2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism

J. J. Liang and P. N. Suganthan

Abstract—**In this paper, a novel constraint-handling mechanism based on multi-swarm is proposed. Different from the existing constraints handling methods, the sub-swarms are adaptively assigned to explore different constraints according to their difficulties. The new mechanism is combined in Dynamic Multi-swarm Optimizer (DMS-PSO) for handling constrained real-parameter optimization problems and Sequential Quadratic Programming (SQP) method is combined to improve its local search ability. The performance of the modified DMS-PSO on the set of benchmark functions provided by CEC2006 [1] is reported.**

I. Introduction

Optimization of constrained problems is an important area in the optimization field since most optimization problems have constraints of different types due to the physical, geometric and other limitations. In general, the constrained problems can be transformed into the following form:

$$\text{Minimize } f(\mathbf{x}), \mathbf{x} = [x_1, x_2, ..., x_D] \quad (1)$$

subjected to:

$$g_i(\mathbf{x}) \le 0, i = 1, ..., q$$
$$h_j(\mathbf{x}) = 0, j = q+1, ..., m$$

$\mathbf{x} \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$. q is the number of inequality constraints and m-q is the number of equality constraints. For the convenience of practice, the equality constraints are always transformed into the inequality form:

$$|h_j(\mathbf{x})| - \varepsilon \le 0$$

where ε is the allowed tolerance. The problem is reformulated as:

$$\text{Minimize } f(\mathbf{x}), \mathbf{x} = [x_1, x_2, ..., x_D] \quad (1)$$

subject to:

$$g_i(\mathbf{x}) \le 0, i = 1, ..., m$$

If we denote with $\mathcal{F}$ the feasible region and $\mathcal{S}$ the whole search space, $\mathbf{x} \in \mathcal{F}$ if $\mathbf{x} \in \mathcal{S}$ and all constraints are satisfied. In this case, $\mathbf{x}$ is called feasible solution.

Evolutionary algorithms have achieved success in various applications. Since they are developed as unconstrained search techniques, when they are used to solve constrained problems, an additional mechanism is required to be incorporated into the fitness function or the evolution strategies to guide the search direction. A variety of approaches are proposed to achieve this objective.

The penalty functions, which were proposed in 1940s and later expanded by many researchers, use the amount of constraint violation to penalize the infeasible solutions and to favor feasible solutions. Although it is the most popular and simple approach, it has some limitations. In order to overcome these limitations, many alternative methods are proposed, such as rejecting infeasible solutions, repairing the infeasible solutions, designing special mutation and crossover operators, etc.[2]. During the last few years, some researchers incorporated the concepts of multi-objective optimization into the constraint-handling techniques. They use Pareto dominance as a selection criterion, assign the nondominated solutions with a higher fitness value or split the population into subpopulations and make each subpopulation focuses on the objective function or one single constraint function [3].

Particle swarm optimizer (PSO) is a new member in the evolutionary optimization area introduced by Kennedy and Eberhart in 1995 [4][5]. The PSO algorithm is easy to implement and has been empirically shown to perform well on many optimization problems. However, solving constrained problems using PSO has not attracted enough attention. The current approaches just combine the existing constraint-handling methods with PSO. In [6], the search was started with a group of feasible solutions and only feasible solutions in the search process were kept; [7] employed a simple penalty function; [8] compared the preservation of feasible solutions method and penalty function method; [9] used subpopulation and use MOPSO in each subpopulation; [10] preferred the feasible solution to the infeasible solution and sorted the feasible solutions according to the objective function while it sorted the infeasible solutions according to the sum of constraints violation.

In this paper, a novel constraint-handling mechanism is proposed. It is similar to the approach used in HCVEGA [3], where each subpopulation has its own objective, chosen from the objective function and constrained functions. Different from the HCVEGA, the objectives are assigned to the subpopulations adaptively and the assignment is periodically changed. The novel constraint-handling mechanism is incorporated into the dynamic multi-swarm particle swarm optimizer (DMS-PSO) [11][12], which has a good global search ability, to solve 24 the constrained problems provided by CEC2006.

The DMS-PSO is introduced in Section II and the novel constrain-handling mechanism is described in Section III. In Section IV, the whole algorithm is summarized and the

The Authors are with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mails: liangjing@pmail.ntu.edu.sg, epnsugan@ntu.edu.sg).

0-7803-9487-9/06/$20.00/©2006 IEEE

experiments results are presented and discussed in Section V.

II. Dynamic Multi-Swarm Particle Swarm Optimizer

In PSO, each potential solution is regarded as a particle. All particles have fitness values and velocities. The particles fly in the D dimensional problem space by learning from the historical information of all the particles. Using the useful information collected in the search process, the particles have a tendency to fly towards better search area over the course of search process. The velocity V_i^d and position X_i^d updates of d^{th} dimension of the i^{th} particle are presented below:

$$V_i^d = w * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (gbest^d - x_i^d) \quad (2)$$

$$X_i^d = X_i^d + V_i^d \quad (3)$$

where c_1 and c_2 are the acceleration constants, $rand1_i^d$ and $rand2_i^d$ are two uniformly distributed random numbers in the range [0,1]. $\mathbf{X}_i = (X_i^1, X_i^2, ..., X_i^D)$ is the position of the i^{th} particle; $\mathbf{pbest}_i = (pbest_i^1, pbest_i^2, ..., pbest_i^D)$ is the best previous position yielding the best fitness value $pbest_i$ for the i^{th} particle; $\mathbf{gbest} = (gbest^1, gbest^2, ..., gbest^D)$ is the best position discovered by the whole population; $\mathbf{V}_i = (v_i^1, v_i^2, ..., v_i^D)$ represents the rate of the position change (velocity) for particle i. w is the inertia weight used to balance between the global and local search abilities.

In the PSO domain, there are two main variants: global PSO and local PSO. In the local version of PSO, each particle's velocity is adjusted 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 as in the global version. The velocity updating equation becomes:

$$V_i^d = w * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (lbest_i^d - x_i^d) \quad (4)$$

where $\mathbf{lbest}_i = (lbest_i^1, lbest_i^2, ..., lbest_i^D)$ is the best position achieved within its neighborhood.

The dynamic multi-swarm particle swarm optimizer is constructed based on the local version of PSO with a new neighborhood topology. In DMS-PSO, the population is periodically and randomly divided into sub-swarms. Each swarm uses its own members to search for better areas in the search space. In this way, the information obtained by each swarm is exchanged among the swarms. This dynamic neighborhood structure has more freedom when compared with the classical neighborhood structure and it has been proved that it can achieve better performance on multimodal problems.

In [12], two concepts are combined into the original DMS-PSO for seeking better performance. The first one is instead of just using the basic updating equation, when updating the positions of the particles, half of the dimensions are kept the same as its best historical position, pbest, to make better use of the particles' historical information to improve its global search ability. The second is that a local search is added into DMS-PSO in order to give a better search in the better local areas. Every L generations, some particles are chosen as the start points to do the local search. The details can be found in [12].

III. The Novel Constraint-Handling Mechanism

The original DMS-PSO is designed to handle unconstrained optimization problems, thus in order to handle constrained problems, a constraint-handling mechanism is required to guide the swarm to search for feasible region. Based on its multi-swarm property, it is natural to let the sub-swarms take charge of different tasks. The constraint-handling approach used in HCVEGA is a good reference, where each subpopulation tries to evolve along one single constraint or the objective function and each individual in a subpopulation is allowed to mate with any other in any subpopulation. By this way, it is expected to have population of feasible individuals with high fitness values, but it has a main drawback that the number of subpopulations needs to increase linearly with the number of constraints of the problem. In order to overcome this weakness, in the novel constraint-handling mechanism, the objective and constraints are assigned to the sub-swarms adaptively according to the difficulties of the constraints.

Suppose that there are m constraints, the population is divided into n sub-swarms with sn members in each sub-swarm and the population size is ps ($ps=n*sn$). n is a positive integer and '$n=m$' is not required.

Define
$$a > b = \begin{cases} 1 & if \quad a > b \\ 0 & if \quad a \le b \end{cases} \quad (5)$$

$$p_i = \frac{\sum_{j=1}^{ps} (g_i(\mathbf{x}_j) > 0)}{ps}, \quad i=1,2,...,m \quad (6)$$

$$fp = 1 - \bar{\mathbf{p}}, \mathbf{p} = [p_1, p_2, ..., p_m] \quad (7)$$

$$gp_i = \bar{\mathbf{p}} \cdot (p_i / \sum_{i=1}^{m} p_i) \quad (8)$$

thus
$$fp + \sum_{i=1}^{m} gp_i = 1 \quad (9)$$

For each sub-swarm, the first step is using roulette selection according to fp and gp_i to assign the objective function or a single constraint as its target. If sub-swarm i is assigned to improve constraint j, set $obj(i)=j$ and if sub-swarm i is assigned to improve the objective function, set $obj(i)=0$. The second step is assigning swarm member for this sub-swarm. Sort the unassigned particles according to $obj(i)$, and assign the best and sn-1 worst particles to sub-swarm i.

While comparing two particles i and j, the following comparison criteria is used:

1. If $obj(i) = obj(j) = k$, particle i wins if

$$g_k(\mathbf{x}_i) < g_k(\mathbf{x}_j)$$
$$\text{or} \quad V(\mathbf{x}_i) < V(\mathbf{x}_j) \ \& \ g_k(\mathbf{x}_i) = g_k(\mathbf{x}_j)$$
$$\text{or} \quad f(\mathbf{x}_i) < f(\mathbf{x}_j) \ \& \ V(\mathbf{x}_i) = V(\mathbf{x}_j) \quad (10)$$

2. If $obj(i) = obj(j) = 0$ or $obj(i) \neq obj(j)$, particle i wins if

$$V(\mathbf{x}_i) < V(\mathbf{x}_j)$$
$$\text{or } f(\mathbf{x}_i) < f(\mathbf{x}_j) \ \&\ V(\mathbf{x}_i) = V(\mathbf{x}_j) \quad (11)$$

where
$$V(\mathbf{x}) = \sum_{i=1}^{m} (weight_i \cdot g_i(\mathbf{x})) \quad (12)$$

$$weight_i = \frac{1/g_i \max}{\sum_{i=1}^{m} (1/g_i \max)} \quad , i=1,2,...m \quad (13)$$

The effect of **weight** is to balance the impacts of different constraints. It is experimentally proven that **weight** is very useful when the differences are too huge among the constraints. g_i max is the estimated maximum value of the constraint i and is updated periodically, thus the estimated g_i max becomes more and more accurate.

In this way, the constraints that are more difficult will have more sub-swarms work for it, while the easier ones will have less or even no sub-swarm working for it. And the search will focus on finding feasible solutions and then concentrate on improving the objective function. There will be more sub-swarms evolve along the fitness increasing direction if more constraints are satisfied.

IV. Dynamic Multi-Swarm Particle Swarm Optimizer with the New Constraint-Handling Mechanism

The DMS-PSO and the novel constraint-handling mechanism have been introduced in Section II and III. The new algorithm which combines DMS-PSO with the new constraint-handling mechanism, which is called DMS-C-PSO, is described as below:

ns: Each swarm's population size *n*: number of swarms
ps: population size, *ps*=*n***ns* *R*: Regrouping period
L: local refining period
FES: fitness evaluations used.
L_FES: Max FEs using in the local search
Max_FEs: Max fitness evaluations, stop criterion

Step 1: Initialization -

Initialize *ps* particles (position **X** and velocity **V**), calculate $f(\mathbf{X})$, $g_j(\mathbf{X})$ (j=1,2...,m) for each particle. Set **pbest**=**X** and initialize $g_i max$ for each constraint. Set *FES*=*ps*.

Step 2: Divide the population into sub-swarms -

Update $g_i max$ for each constraint and calculate **weight** according to (13).

Calculate *fp* and **gp** according to (7) and (8).

For each sub-swarm *i*, assign *obj(i)* using roulette selection according to *fp* and **gp**. Then sorting the unassigned particles according to *obj(i)* and assigning the best and *sn*-1 worst particles to sub-swarm *i*.

ns can be a small integer since PSO needs a comparatively smaller population size when compared to other EAs. PSO with small neighborhoods might perform better on complex problems [13]. *ns*=3 is a reasonable choice [11].

Step 3: Update the particles -

For each particle, find its local best **lbest**$_i$ and update the position $\mathbf{X}_i$ and velocity $\mathbf{V}_i$ as below:

For d=1:D
  If rand<0.5
$$V_i^d = w * V_i^d + c_1 * rand1_i^d * (pbest_i^d - X_i^d) + c_2 * rand2_i^d * (lbest_i^d - x_i^d)$$
$$V_i^d = \min(\max(V_i^d, -V_{\max}^d), V_{\max}^d)$$
$$X_i^d = X_i^d + V_i^d$$
  Else $X_i^d = pbest_i^d$
  End
End

If $\mathbf{X}_i \in [\mathbf{X}_{\min}, \mathbf{X}_{\max}]^D$, calculate $f(\mathbf{X}_i)$, $g_j(\mathbf{X}_i)$ (j=1,2...,m), update *FES*. Compare $\mathbf{X}_i$ and **pbest**$_i$ using (10) and (11) introduced in Section III. If $\mathbf{X}_i$ wins, update **pbest**$_i$.

Step 4: Regroup -

Every *R* generations, go to Step 2.

Step 5: Local Search -

Every *L* generations, randomly choose 5 particles' **pbest** and start local search with Sequential Quadratic Programming (SQP) method using these solutions as start points. (In the experiments, *fmincon* function in Matlab 6.5 was employed) And *L_FES* is set as the Max *FES* for each local search. Update overall *FES* counter. Compare the result **x** with the start point **pbest**$_i$. If **x** wins, update that **pbest**$_i$.

Step 6: If *FES*≤0.7**Max_FES*, go to Step 3. Otherwise go to Step 7.

Step 7: Set n=1, *ns*=*ps*, *obj*=0, continue search using one swarm. Every *L* generations, start local search using **gbest** as start points using 5**L_FES* as the Max FEs. Stop search if *FES*≥*Max_FES*

V. Experiments

PC Configuration:
System: Windows XP SP1; CPU: 3.00GHz
RAM: 2.00 GB; Language: Matlab 6.5.1
Algorithm: DMS-C-PSO

Parameters Setting:
a) Parameters to be adjusted:
ω, c_1, c_2, **Vmax**, n, *ns*, *R*, *L*, *L_FES*
b) Corresponding dynamic ranges:
All parameters are fixed except n.
c) Guidelines on how to adjust the parameters
n can be small for unimodal problems and large for multimodal problems. If you do not know the property of the problem, 20 is a recommended choice.
d) Actual parameter values used.
ω=0.729, c_1=c_2=1.49445; **Vmax**=0.5*(**Xmax-Xmin**)
n=20; *ns*=3; *R*=100; *L*=500
L_FES=1,000; *Max_FES*=500,000

Experiments are conducted on the 24 constraint real parameter problems provided in [1]. For each problem, the DMS-C-PSO is run 25 times. Best functions error values achieved when FEs=5e+3, FEs=5e+4, FEs=5e+5 and Success Performance and Successful FEs for the 24 test functions are presented in Tables I-V and the computation complexity is given in Table VI. Success here means a feasible solution within the desired error accuracy is found within the permitted number of FEs. The predefined tolerance value for the 24 test functions is 1e-4.

The convergence graphs for the median run of the 24 problems are plotted in Figs. 1-4. And the Computational Complexity [1] is reported in Table V.

From the results, we can observe that among those 24 test functions, g08, g11 and g12 are comparatively easy, and the success performances of DMS-C-PSO are under 5000 while g02, g13, g17, g20 and g21 are comparatively difficult. DMS-C-PSO found feasible solutions for all problems except g20, and achieved 100% success rate for most problems except g2, g17, g20 and g22. From the constraint violation convergence graphs, DMS-C-PSO found feasible solutions efficiently except for g20 and g22. Comparing with the previous results provided in [3], it is obvious that the novel constraint handling mechanism performs very well on finding feasible regions. g2 and g17 are complex multimodal cases. The local optima mislead the particles to search along the false directions. Thus on these problems, a larger diversity is favored, increasing diversity can improve the results but will affect the convergence speed for other problems. There is no feasible solution for g20. g22 is an interesting problem and a feasible solution can be found only when the algorithm employs an elitist strategy. It is why DMS-C-PSO always found good solution at the end of the search, when all the particles are grouped into one swarm and fly to gbest.

TABLE I

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 1-6

FES \ Prob.		g01	g02	g03	g04	g05	g06
5×10^3	Best	1.2630e+001(0)	3.2907e-001(0)	5.4701e-001 (0)	9.9407e+001(0)	4.5181e+002(3)	6.4997e+001(0)
	Median	9.5671e+000(3)	4.4800e-001(0)	8.4346e-001 (0)	3.0653e+002(0)	1.8135e+002(3)	5.5469e+002(0)
	Worst	1.5462e+000(6)	5.0651e-001(0)	9.5638e-001 (0)	5.5921e+002(0)	2.7253e+002(3)	1.5580e+003(0)
	c	2 3 3	0 0 0	0 0 0	0 0 0	3 3 3	0 0 0
	$\bar{v}$	2.9845e-001	0	0	0	1.5135e+001	0
	Mean	9.0738e+000	4.4632e-001	8.1589e-001	3.0772e+002	2.4135e+002	6.6661e+002
	Std	3.1128e+000	3.5419e-002	1.0937e-001	1.2503e+002	3.9524e+002	4.6926e+002
5×10^4	Best	1.3284e-011(0)	1.1011e-002(0)	0(0)	3.2996e-009 (0)	5.6674e-008 (0)	3.5462e-007 (0)
	Median	2.2876e-011(0)	2.9190e-002(0)	0(0)	1.0074e-008 (0)	8.4345e-008 (0)	4.6018e-007 (0)
	Worst	2.3460e-011(0)	5.7426e-002(0)	0(0)	3.5479e-007 (0)	1.4356e-007 (0)	5.4144e-007 (0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	2.1479e-011	3.0004e-002	0	2.3919e-008	8.5780e-008	4.5368e-007
	Std	2.9160e-012	1.4421e-002	0	6.9096e-008	1.8264e-008	4.5740e-008
5×10^5	Best	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Median	0(0)	0 (0)	0(0)	0(0)	0(0)	0(0)
	Worst	0(0)	1.8352e-002(0)	0(0)	0(0)	0 (0)	0(0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\bar{v}$	0	0	0	0	0	0
	Mean	0	1.8910e-003	0	0	0	0
	Std	0	4.6953e-003	0	0	0	0

* c is the number of violated constraints at the median solution: the sequence of three numbers indicate the number of violations (including inequality and equalities) by more than 1.0, more than 0.01 and more than 0.0001 respectively. $\bar{v}$ is the mean value of the violations of all constraints at the median solution. The numbers in the parenthesis after the fitness value of the best, median, worst solution are the number of constraints which cannot satisfy feasible condition at the best, median and worst solutions respectively. Due to the accuracy of the provided $f(\boldsymbol{x}^*)$, when $(f(\boldsymbol{x})-f(\boldsymbol{x}^*))$<1e-10 the final errors are reported as 0.

TABLE II

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 7-12

FES \ Prob.		g07	g08	g09	g10	g11	g12
5×10^3	Best	2.1675e+001(0)	0(0)	7.3774e+000(0)	4.9707e+003(0)	1.3867e-005	3.9803e-009(0)
	Median	1.0535e+002(0)	1.2354e-006 (0)	3.1652e+001(0)	5.0248e+003(1)	4.7784e-004 (0)	1.1441e-004(0)
	Worst	1.1911e+003(1)	1.7851e-002 (0)	1.4446e+002(0)	-4.5229e+002(2)	1.1871e-002 (0)	1.3724e-002(0)
	c	0 0 0	0 0 0	0 0 0	0 1 1	0 0 0	0 0 0
	$\overline{v}$	0	0	0	1.5579e-002	0	0
	Mean	1.4801e+002	1.6177e-003	4.0727e+001	6.9441e+003	2.6798e-003	2.5467e-003
	Std	2.3111e+002	4.2030e-003	3.4893e+001	3.7619e+003	3.5673e-003	4.1882e-003
5×10^4	Best	0(0)	0(0)	0(0)	8.4093e-008 (0)	0 (0)	0(0)
	Median	0(0)	0(0)	0(0)	1.1546e-007 (0)	1.4504e-007 (0)	0(0)
	Worst	0 (0)	0 (0)	0 (0)	1.4053e-007 (0)	2.6982e-005 (0)	0(0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\overline{v}$	0	0	0	0	0	0
	Mean	0	0	0	1.1714e-007	5.1870e-006	0
	Std	0	0	0	1.4908e-008	8.1559e-006	0
5×10^5	Best	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
	Median	0(0)	0(0)	0(0)	1.0124e-008 (0)	0(0)	0(0)
	Worst	0 (0)	0 (0)	0 (0)	3.4056e-008 (0)	0 (0)	0(0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\overline{v}$	0	0	0	0	0	0
	Mean	0	0	0	1.1471e-008	0	0
	Std	0	0	0	8.8141e-009	0	0

TABLE III

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 13-18

FES \ Prob.		g13	g14	g15	g16	g17	g18
5×10^3	Best	9.4602e-001(3)	-5.1638e+001(3)	5.4648e+000(2)	1.0120e-001(0)	8.1153e+001(4)	2.8452e+000(11)
	Median	2.5643e-001(3)	-1.4043e+002(3)	-4.8440e-002(2)	3.8176e-001(0)	3.7268e+002(4)	3.9834e+000(10)
	Worst	4.5701e-001(3)	-2.8691e+002(3)	2.3577e+000(2)	5.1611e-001(1)	-1.3396e+002(4)	-2.6003e+000(9)
	c	0 3 3	3 3 3	0 2 2	0 0 0	4 4 4	9 10 10
	$\overline{v}$	1.8708e-001	5.0018e+000	1.5864e-001	0	2.1883e+001	8.5302e+000
	Mean	1.1045e+000	-1.5107e+002	3.5843e+000	3.9333e-001	1.0229e+002	-2.5120e+000
	Std	1.3503e+000	4.5625e+001	2.5916e+000	1.9453e-001	9.7335e+001	5.6103e+000
5×10^4	Best	0(0)	0(0)	0(0)	3.6299e-000(0)	7.4058e+001(0)	0(0)
	Median	7.4414e-006(0)	0(0)	0(0)	3.3737e-008(0)	7.4058e+001(0)	0(0)
	Worst	3.8489e-001(0)	0 (0)	5.4987e-005 (0)	1.8409e-002(0)	7.4058e+001(0)	1.9104e-001(0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\overline{v}$	0	0	0	0	0	0
	Mean	3.0795e-002	0	2.1996e-006	3.9690e-003	7.4058e+001	1.5300e-002
	Std	1.0657e-001	0	1.0997e-005	6.1081e-003	0	5.2893e-002
5×10^5	Best	0(0)	0(0)	0(0)	0(0)	7.4058e+001(0)	0(0)
	Median	0(0)	0(0)	0(0)	0(0)	7.4058e+001(0)	0(0)
	Worst	7.5911e-006 (0)	0 (0)	0 (0)	0 (0)	7.4058e+001(0)	0 (0)
	c	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
	$\overline{v}$	0	0	0	0	0	0
	Mean	5.1763e-007	0	0	0	7.4058e+001	0
	Std	1.8204e-006	0	0	0	0	0

TABLE IV

ERROR VALUES ACHIEVED WHEN FES= 5×10^3 , FES= 5×10^4, FES= 5×10^5 FOR PROBLEMS 7-12

FES \ Prob.		g19	g20	g21	g22	g23	g24
5×10^3	Best	2.3787e+002(0)	1.2826e+001(19)	2.8574e+002(5)	7.5081e+003(19)	2.1993e+002(5)	5.7750e-004 (0)
	Median	3.6297e+002(0)	6.1567e+000(20)	7.4984e+002(5)	7.8469e+003(19)	-1.9283e+002(5)	8.3913e-003 (0)
	Worst	5.6396e+002(0)	1.4496e+001(20)	7.3949e+002(5)	1.0306e+004(19)	7.7198e+002(6)	9.5512e-002 (0)
	c	0 0 0	2 20 20	2 5 5	18 19 19	3 5 5	0 0 0
	$\overline{v}$	0	4.5795e+000	2.0254e+000	6.1988e+007	1.9762e+000	0
	Mean	3.6710e+002	9.1305e+000	4.4678e+002	1.0213e+004	-1.2366e+002	1.6628e-002
	Std	8.0830e+001	2.3124e+000	2.0062e+002	5.5033e+003	4.9379e+002	2.3570e-002
5×10^4	Best	9.2910e-009(0)	-7.3850e-002(15)	2.9533e-006(0)	7.2813e+000 (5)	2.2999e-003(0)	0(0)
	Median	1.9522e-007(0)	-7.7915e-002(15)	3.3558e-002(0)	1.2674e+004(18)	1.0878e-002(0)	0(0)
	Worst	2.7958e-006(0)	-9.0246e-002(11)	8.5295e+001(0)	3.3874e+003(20)	1.1210e-001(0)	0 (0)
	c	0 0 0	0 8 15	0 0 0	4 6 18	0 0 0	0 0 0
	$\overline{v}$	0	2.8005e-002	0	1.2781e+001	0	0
	Mean	4.3410e-007	-7.4462e-002	7.5329e+000	8.4038e+003	3.5801e-002	0
	Std	6.2999e-007	8.1726e-003	2.2467e+001	6.8442e+003	4.4219e-002	0
5×10^5	Best	0(0)	-7.3886e-002(16)	8.7143e-007(0)	2.5658e+000 (0)	5.7418e-009 (0)	0(0)
	Median	0(0)	-7.3330e-002(17)	3.5911e-006(0)	1.2200e+002 (0)	1.0267e-008 (0)	0(0)
	Worst	0 (0)	-5.7810e-002(16)	1.0693e-005(0)	1.2200e+002 (0)	2.2999e-003 (0)	0 (0)
	c	0 0 0	0 6 17	0 0 0	0 0 0	0 0 0	0 0 0
	$\overline{v}$	0	2.6113e-002	0	0	0	0
	Mean	0	-6.7860e-002	4.0328e-006	3.8068e+001	9.2005e-005	0
	Std	0	6.9516e-003	2.0784e-006	2.7703e+001	4.5997e-004	0

TABLE VI

NUMBER OF FES TO ACHIEVE THE FIXED ACCURACY LEVEL (($f(\vec{x})$ - $f(\vec{x}^*)$) ≤ 0.0001)

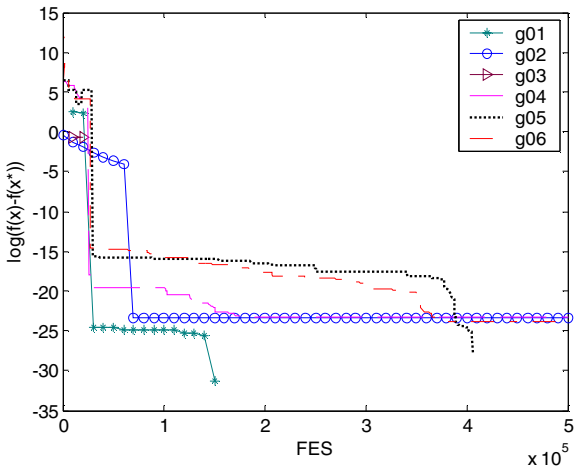
SUCCESS RATE, FEASIBLE RATE AND SUCCESS PERFORMANCE

Prob.	Best	Median	Worst	Mean	Std	Feasible Rate	Success Rate	Success Performance [1]
g01	22844	25816	47816	33333	10887	100%	100%	33333
g02	53379	87107	5e+005	1.7513e+005	1.6027e+005	100%	84%	1.7513e+005
g03	24500	25500	26500	25580	471.7	100%	100%	25580
g04	24974	25443	25777	25404	207.75	100%	100%	25404
g05	28500	29000	31000	29380	545.44	100%	100%	29380
g06	26656	27636	28287	27636	455.44	100%	100%	27636
g07	25574	26685	27416	26578	439.13	100%	100%	26578
g08	1621	3892	7990	4124.4	1727	100%	100%	4124.4
g09	29272	29410	29782	29456	138.29	100%	100%	29456
g10	24500	25500	26500	25520	549.24	100%	100%	25520
g11	963	12921	29362	14625	11973	100%	100%	14625
g12	812	6826	9192	5409.4	2841.9	100%	100%	5409.4
g13	28350	30086	2.2897e+005	41090	42719	100%	100%	41090
g14	22500	24000	50500	25220	5383.4	100%	100%	25220
g15	28500	29000	30000	28900	353.55	100%	100%	28900
g16	26671	28433	1.1309e+005	53480	33328	100%	100%	53480
g17	-	-	-	-	-	100%	0%	-
g18	27500	28000	90500	33180	14814	100%	100%	33180
g19	20644	21587	24721	21830	905.45	100%	100%	21830
g20	-	-	-	-	-	0%	0%	-
g21	27000	1.485e+005	2.67e+005	1.4034e+005	71337	100%	100%	1.4034e+005
g22	-	-	-	-	-	100%	0%	-
g23	54000	1.95e+005	5e+005	2.1402e+005	1.0498e+005	100%	100%	2.1053e+005
g24	13721	18729	26096	19376	3383	100%	100%	19376

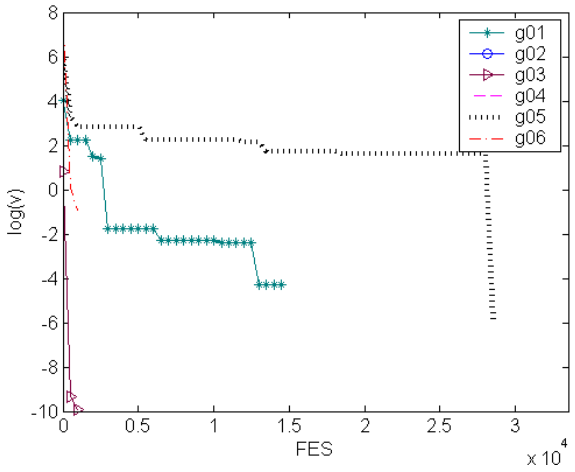
TABLE VI

COMPUTATIONAL COMPLEXITY

T1	*T2*	*(T2-T1)/T1*
5.1162	9.9315	1.0581

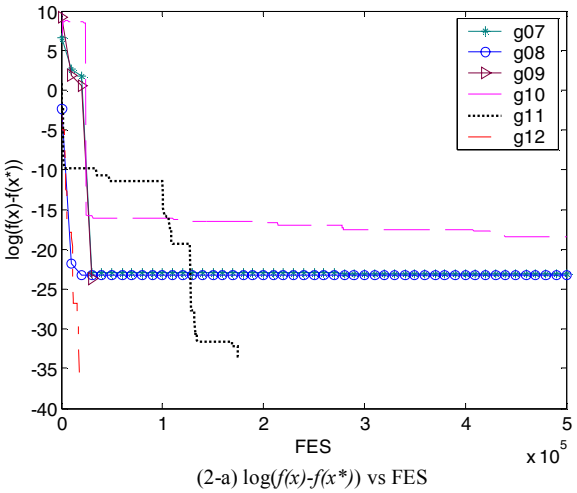


(1-a) log(*f(x)-f(x*)*) vs FES

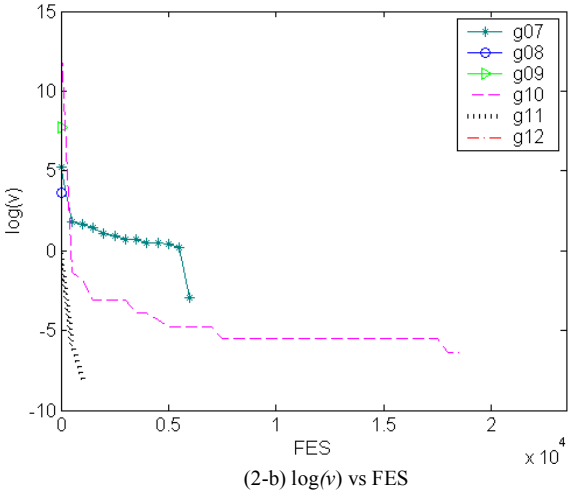


(1-b) log(*v*) vs FES

Fig. 1: Convergence Graph for Function 1-6

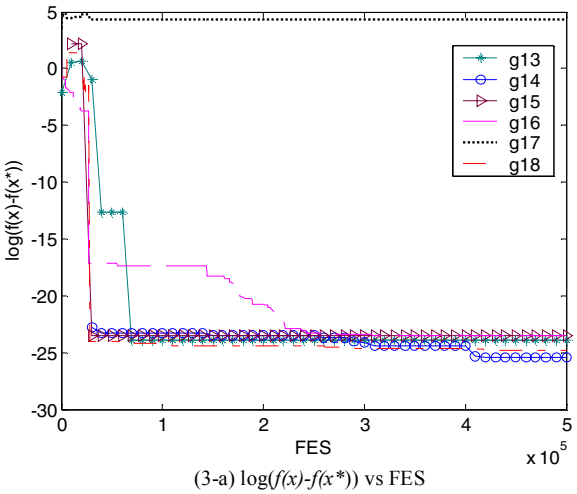


(2-a) log(*f(x)-f(x*)*) vs FES

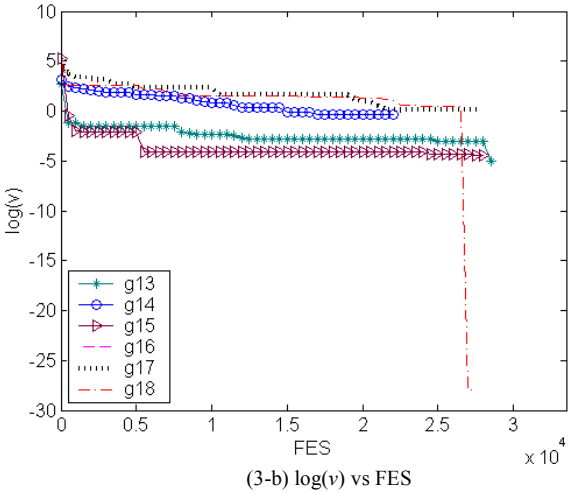


(2-b) log(*v*) vs FES

Fig. 2: Convergence Graph for Function 7-12

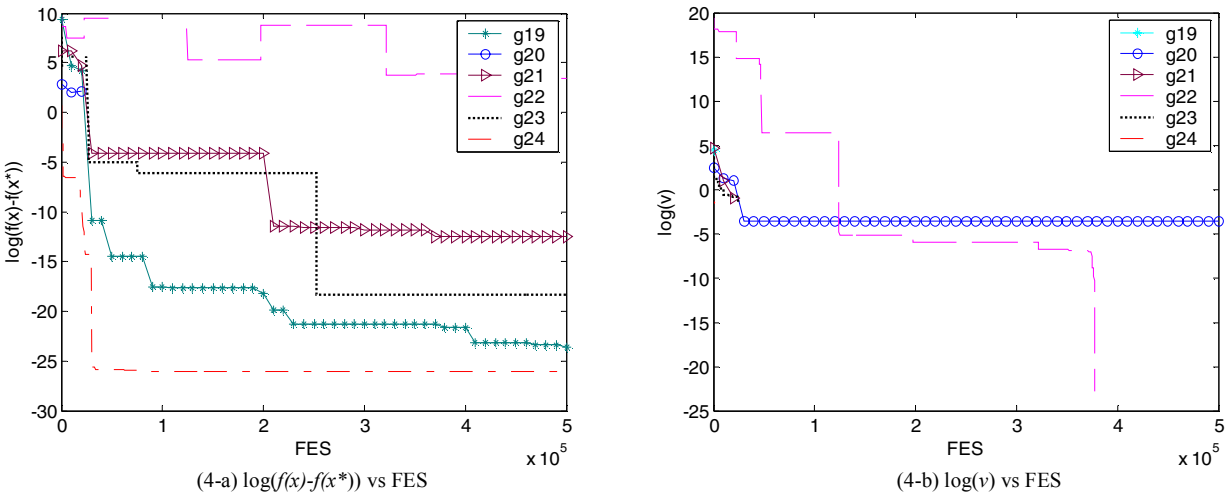


(3-a) log(*f(x)-f(x*)*) vs FES



(3-b) log(*v*) vs FES

Fig. 3: Convergence Graph for Function 13-18



(4-a) log(*f(x)-f(x**)) vs FES (4-b) log(*v*) vs FES

Fig. 4: Convergence Graph for Function 13-18

VI. Conclusion

In this paper, a novel constraint-handling mechanism based on multi-swarm is proposed to be employed in the Dynamic Multi-swarm Optimizer (DMS-PSO) to handle constrained real-parameter optimization problems. With the new constraint-handling mechanism, the sub-swarms are adaptively assigned to explore different constraints in the search process. A local search is combined with the algorithm to improve its local search ability. The modified DMS-PSO with local search is tested on a set of benchmark functions and the results show that the proposed algorithm can find feasible region for all test problems that have a feasible region and find the global optimum for most problems. For the comparison of DMS-C-PSO with other algorithms, please refer to other articles in this special session.

References

[1] J. J. Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, P. N. Suganthan, Carlos A. Coello Coello & K. Deb,"Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization", *Technical Report*, Nanyang Technological University, Singapore, Dec 2005.

[2] S. Koziel and Z. Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization". *Evolutionary Computation*, pp. 19-44,Volume 7, Number 1, 1999

[3] E. Mezura-Montes and C. A. C. Coello, "A Numerical Comparison of some Multiobjective-Based Techniques to Handle Constraints in Genetic Algorithms", *Technical Report EVOCINV-0302002*, Evolutionary Computation Group at CINVESTAV, México, 2002.

[4] R. C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory", *Proc. of the Sixth Int. Symposium on Micromachine and Human Science*, Nagoya, Japan. pp. 39-43, 1995

[5] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization ". *Proc.s of IEEE International Conference on Neural Networks*, Piscataway, NJ. pp. 1942-1948, 1995

[6] X. Hu and R. C. Eberhart," Solving constrained nonlinear optimization problems with particle swarm optimization". *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002)*, Orlando, USA. 2002

[7] N. Li, F. Liu, D. Sun, and C. Huang, "Particle Swarm Optimization for Constrained Layout Optimization" *Proc. of the Fifth World Congress on Intelligent Control and Automation*, pp: 2214 - 2218 Vol.3, 2004

[8] G. Coath and S. K. Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems". *Proc. of IEEE Congress on Evolutionary Computation 2003 (CEC 2003*), Canbella, Australia. pp. 2419-2425, 2003

[9] J. Chunlin, "A Revised Particle Swarm Optimization Approach for Multi-Objective and Multi-Constraint Optimization*", Proc. GECCO 2004 Conference*, 2004.

[10] G. T. Pulido and C. A. Coello Coello, "A Constraint-Handling Mechanism for Particle Swarm Optimization", *Proc. of IEEE Congress on Evolutionary Computation (CEC'2004)*, pp. 1396-1403, Vol. 2, IEEE, Portland, Oregon, June 2004.

[11] J. J. Liang, and P.N. Suganthan,"Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search" *Proc. of IEEE Congress on Evolutionary Computation (CEC 2005),* pp.522 - 528 Vol. 1, 02-05, Sept. 2005

[12] J. J. Liang, and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer," *Proc. of IEEE International Swarm Intelligence Symposium*, pp. 124-129, 2005

[13] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance ". *Proc. of IEEE Congress on Evolutionary Computation (CEC 1999)*, Piscataway, NJ. pp. 1931-1938, 1999