ISSN 0974 - 9373

Vol. 18 No.2 (2014) **Journal of International Academy of Physical Sciences** pp. 151-165

An Effective Hybrid DE with PSO for Constrained Engineering Design Problem

Raghav Prasad Parouha and Kedar Nath Das
Department of Mathematics
National Institute of Technology Silchar, Assam
Email: rparouha@gmail.com; kedar.iitr@gmail.com

(Received September 25, 2013)

Abstract: Many engineering design problems can be formulated as constrained optimization problems. Solving constrained engineering design problems via evolutionary algorithms has attracted increasing attention in the past decade. So far, penalty function methods have been the most popular methods for constrained optimization due to their simplicity and easy implementation. However, it is often not easy to set suitable penalty factors. This paper proposes an alternative hybrid approach (namely DPD) to efficient solving for constrained engineering design optimization problems combining by differential evolution (DE) and particle swarm optimization (PSO) algorithms. DPD is based on tri-break-up concept of population. Initially all individual in the population are divided into three groups – Inferior Group, Mid Group and Superior Group; according to their increasing order of function value. Initially the suitable mutation operators for both DEs used in DPD are investigated. The investigated mutation combination for DEs in DPD algorithm is shown to enhance the local search ability of the basic DE. Moreover, two strategies Elitism and Non-redundant search improve the quality of the solutions in the memory of each individual. Under the guidance of the bracket operator penalty (exterior penalty), the algorithm quickly finds better feasible solution. This algorithm has been applied to two constrained engineering optimization problems reported in the specialized literature. DPD compared with respect to algorithms representative of the state-of-the-art in the area. The results indicate that the DPD is a powerful optimization technique that may yield better solutions to engineering problems.

Keywords: Differential Evolution; Particle Swarm Optimization; Non-redundant search; Elitism; Engineering design problem

1. Introduction

Nowadays, economy plays an important role in all aspects of human life. Since engineering projects are time, energy, and cost consuming. Economy has a great influence on them and optimization is an inevitable part of

engineering design problem. Many engineering design problems can be formulated as constrained optimization problems, which involve a number of constraints that the solutions had to satisfy. Generally, a constrained optimization problem can be described as follows:

$$Minimize / Maximize \;\; f(X),\; X = \{x_1, x_2, x_3, ..., x_n\} \in R^n$$

Subject to

$$g_i(X) \geq 0; \qquad i = 1, 2, ..., p$$
$$h_j(X) = 0; \qquad j = 1, 2, ..., m$$
$$a_d^L \leq x_i \leq b_d^L; \quad d = 1, 2, ..., n$$

where $X = \{x_1, x_2, x_3, ..., x_n\}$ is an n dimensional position vector, $f(X)$ is an objective function, $g_i(X) \geq 0$ is p inequality constraints, $h_i(X) = 0$ is m equality constraints respectively. Values a_d^L and b_d^L are the lower and upper boundaries of x_i respectively. As equality constraints are difficult to be tackled in the optimization problems. All equality constraints $h_j(x) = 0$ could be broken into two inequalities with ≤ and ≥ sign. Again ≤ sign can be modified into ≥ sing by multiplying (-1), both sides.

In recent years, many evolutionary algorithms (EAs) have been proposed for solving constrained engineering optimization problems. EAs possess a number of exclusive advantages: generality, reliable and robust performance, little information requirement for the problem to be solved, easy implementation, etc. Especially, Differential Evolution (DE)[1] and Particle Swarm Optimization (PSO)[2] are two formidable population-based optimizers that follow different philosophies and paradigms, which are successfully and widely applied in scientific and engineering research. Because of the individual shortcomings of each of DE and PSO, the solution leads to a premature convergence or getting stack in some local optima. The "no free lunch theorem[3]" states that no individual optimization algorithm is better than all the other optimization algorithms for all classes of optimization problems. However, from the literature, there is no single EA algorithm which is able to solve a wide range of constrained optimization problems consistently. Hence, researchers started working on the hybridization techniques between DE and PSO in order to maintain a proper balance between exploration and exploitation in the search space. The hybridization between DE and PSO represents a promising way to create more powerful optimizers, especially for specific problem solving.

In past decade, many hybrid algorithms have been proposed for solving constrained engineering optimization problems. Liu et al.[4] used a hybrid

PSO called PSO-DE to solve constrained numerical and engineering optimization problems. In PSO-DE, DE is incorporated to update the previous best positions of PSO particles to force them to jump out of local attractor in order to prevent stagnation of population. The PSO-DE integrates PSO with differential evolution (DE) to obtain a good performance. Khamsawang et al.[5] proposed an improved hybrid algorithm based on conventional particle swarm optimization and differential evolution (called PSO-DE) for solving an economic dispatch (ED) problem with the generator constraints.

Recently, Nwankwor et al.[6] proposed a hybrid version of PSO and DE for the optimal well placement problem. The combination of PSO and DE seems to be very effective as suggested in the report by Xin et al.[7]. The said approach moves around the enhancement of capabilities of PSO and DE in various aspects. Yadav and Deep[8] proposed a new co-swarm PSO (CSHPSO) for constrained optimization problems, which is obtained by hybridizing the recently proposed shrinking hypersphere PSO (SHPSO) with the differential evolution (DE) approach. The total swarm is subdivided into two sub swarms in such a way that the first sub swarms uses SHPSO and second sub swarms uses DE. CSHPSO apply in benchmark problems, power system optimization problem with valve point effects. The results demonstrate that the proposed CSHPSO algorithm shows better performance in comparison to the state-of-the-art algorithms. It can be concluded that the hybridization of DE and PSO came out as a giant optimizer for the optimization problems.

In view of the above problems, this paper puts forward a new algorithm hybridizing differential evolution (DE) with particle swarm optimization (PSO). DPD combines DE with PSO on the basis of an optimal information sharing mechanism firstly. In this study, the process of hybridization is being emphasized with a tri-breakup of the population. The novel hybrid algorithm thus proposed is named as DE-PSO-DE for solving constrained engineering optimization problems.

The rest of the paper is organized as follows. Section II introduces the different components of the proposed hybrid system. Section III represents the proposed algorithm. The experimental setting is reported in section IV. Section V contains the results and discussions. The Conclusion and future works is drawn at the last section VI.

2. The Framework of Proposed Hybrid Approach

Constrained Handling Technique

An efficient and adequate constraint-handling technique is a key element in the design of optimization algorithm. Due to the simplicity, the penalty

function method has been considered as the most popular technique to handle problem-specific constraints. Although the use of penalty functions is the most common technique for constraint-handling. Use of penalty functions has been commonly reported in literature for use in constrained optimization. Michalewicz[9] summarized several constraint-handling techniques, and it was pointed out that the penalty function is the most widely used technique to handle constraints due to its simple principle and easy implementation. Two basic types of penalty functions exist; exterior penalty functions, which penalize infeasible solutions, and interior penalty functions, which penalize feasible solutions. In the EAs community exterior Penalty function method is preferred to the interior Penalty function approach. The reason is that interior function approach requires an initial feasible solution, which is the main drawback of this method. Exterior Penalty function method starts with infeasible or feasible solution. If it starts with infeasible solution then penalty function forces the algorithm to move towards the feasible solution and if it starts with feasible solution the penalty function becomes zero so it does not affect the value of the objective function. The main advantage of the use of penalty functions is their simplicity; however, their main shortcoming is that penalty function methods require the fine-tuning of the penalty function parameters, to discourage premature convergence, whilst maintaining an emphasis on optimality. There are various forms of penalties reported in the literature, like parabolic penalty, Infinite barrier penalty, Log penalty, and Inverse penalty, Bracket operator penalty. Overall, Penalty function methods are simple and convenient, which don't strictly require problem itself, but how to determine the suitable penalty factors is a tough problem. At present, there has not been a generally effective method for solving constrained optimization problem.

In this paper, fitness function of each individual in the population find by using the bracket operator penalty (exterior penalty). This penalty is mostly used in handling inequality constraints[10]. The constrained optimization problem transformed into unconstrained problem is of the form

$$F(x^{(t)}, R^{(t)}) = f(x^{(t)}) + \Omega\,(R^{(t)}, g^{(t)}, h^{(t)}),$$

where, F stands for the objective function and Ω stands for penalty term which is given by

$$\Omega = \mathrm{R}\left\langle g(\mathrm{x})\right\rangle^2 \qquad \text{where } \left\langle \alpha \right\rangle = \begin{cases} 0, & \text{if } \alpha \geq 0 \\ \alpha, & \text{if } \alpha < 0 \end{cases}.$$

Differential Evolution (DE)

Differential Evolution (DE) is a population-based optimization algorithm which is similar to the genetic algorithm. But the sequence of applying them is different. DE works with three major steps such as Mutation, Crossover and Selection.

(a) Mutation: It is a central/core operator in DE. There are many mutation strategies in DE. Some well-known mutation operators are listed as follows:

1) "DE/rand/1/bin"

$$V_i = x_{r_1} + F \times (x_{r_2} - x_{r_3}),$$

2) "DE/rand/2/bin"

$$V_i = x_{r_1} + F \times (x_{r_2} - x_{r_3}) + F \times (x_{r_4} - x_{r_5}),$$

3) "DE/best/1/bin"

$$V_i = x_{best} + F \times (x_{r_2} - x_{r_3}),$$

4) "DE/best/2/bin"

$$V_i = x_{best} + F \times (x_{r_2} - x_{r_3}) + F \times (x_{r_4} - x_{r_5}),$$

5) "DE/rand-to-best/1/bin"

$$V_i = x_{r_1} + F \times (x_{best} - x_{r_1}) + F \times (x_{r_2} - x_{r_3}),$$

6) "DE/current-to-best"

$$V_i = x_i + F \times (x_{best} - x_i) + F \times (x_{r_2} - x_{r_3}),$$

7) "DE/rand-to-best/2/bin"

$$V_i = x_{r_1} + F \times (x_{best} - x_{r_1}) + F \times (x_{r_2} - x_{r_3}) + F \times (x_{r_4} - x_{r_5}),$$

8) "DE/current-to-best/2/bin"

$$V_i = x_i + F \times (x_{best} - x_i) + F \times (x_{r_2} - x_{r_3}) + F \times (x_{r_4} - x_{r_5}),$$

where a random number $F \in [0,1]$ is the mutation coefficient, x_{best} represents the best individual in the current generation $i \neq r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq r_6 \in \{1,....,N_p\}$ x_i is referred to the target vector; v_i is the mutant vector.

(b) Crossover: After mutant vectors generated, DE introduce the crossover operation which increases the diversity of the target vectors. Then crossover operation, generates the population of candidates as follows:

$$U_{j,i,G+1} = \begin{cases} V_{j,i,G+1}; \text{ if } (rand_j \leq CR) \text{ or } (j = j_{rand}) \\ X_{j,i,G}\ ; \text{ if } (rand_j > CR) \text{ or } (j \neq j_{rand}) \end{cases},$$

where $j=1, 2,..,D;$ $rand_j \in [0,1]$; CR is the crossover constant takes values in the range [0,1] and $j_{rand} \in (1,2,.., D)$ is the randomly chosen index.

(c) Selection:- The vector having better fitness value is then selected as a promising individual for the next generation according to the following rule:

$$x_{i,G+1} = \begin{cases} U_{i,G+1;} & \text{if} \quad f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G}; & \text{otherwise} \end{cases}.$$

Particle Swarm Optimization (PSO)

PSO is a robust stochastic optimization technique based on the movement and intelligence of swarms. PSO also starts with an initial population with the population size. Each individual in the population is called particle that involves with a position *x* and velocity *v*. The *x* and *v* of the i^{th} particle are respectively given as $x_i = (x_{i1}, x_{i2}, ..., x_{iN})$ and $v_i = (v_{i1}, v_{i2}, ..., v_{iN})$ where *N* stands for the dimensions of the problem. During the evolution, each particle flies to its previous best position *pBest* and the global best position *g Best* found so far. Hence, a particle's velocity and position are updated as follows:

$$v = w.v + c_1 r_1 (pBest - x) + c_2 r_2 (gBest - x),$$

$$x_{iN}(t+1) = x_{iN}(t) + v_i(t+1),$$

where ω is the "inertia weight", c_1 and c_2 are positive constants, "acceleration coefficients", r_1 and r_2 are random numbers that are uniformly distributed in the interval [0, 1].

Elitism

If crossover or mutation performed in an evolutionary algorithm (EA) then good candidates may be lost in offspring that are weaker than the parents. Often the EA will re-discover these lost improvements in a subsequent generation but there is no guarantee. To combat this we can use a feature known as elitism. Elitism is therefore is a mechanism to retain the overall best individuals. At the end of iteration both the populations (obtained before and after the iterations) are combined and the best half is considered for next generation.

NRS (Non-Redundant Search)

In proposed algorithm, Non-redundant search (NRS)[11] is used as a local search. The NRS algorithm works as follows:

a. Deletion of individuals with the same chromosome in the current population.
b. Addition of new individuals selected randomly instead of these redundant ones.

Consequently, non-redundant search improves the search ability to find the optimal solution.

3. Proposed Method DE-PSO-DE

Inspiration:

Han,et. al.[12] developed a dynamic group-based differential evolution (GDE) algorithm for global optimization problems, which has both exploitation and exploration abilities. The GDE algorithm provides a generalized evolution process based on two mutation operations to enhance search capability. In GDE algorithm, initially all individuals in the population are grouped into a superior group and an inferior group based on their fitness. The two groups perform different mutation operations. The local mutation model is applied in the superior group, to search for better solutions near the current best position. The global mutation model is applied to the inferior group, which is composed of individuals with lower fitness values, to search for potential solutions. The GDE algorithm performs two mutation operations based on different groupings to effectively search for the optimal solution.

Observation/Motivation of DPD:

Particle swarm optimization (PSO) and Differential evolution scheme is the contemporary, both swarm intelligence and evolutionary process. The motivation behind the hybridization of DE and is PSO to take advantage for providing better solution, simultaneously[4-8]. Due to the robust behavior of mutation operators, DE has the ability to balance exploration and exploitation, over search space. As. Han,et. al.[12], DE works better at two different situations

where the local search or the global search is essential. However, due to the inherent shortcomings of DE, sometimes, stacking in some local minimal or choosing the path to premature convergence is unavoidable. Hence many a time the diversity in the population need to be maintained. Therefore introduction of another mechanism becomes essential in the group based hybridization methods, proposed in. Han et al.[12]. It is also observed that the behavior of PSO is to wildly seek the potential solution. It diversifies the candidate solutions in a better way and probably also helps in avoiding some shortcomings of DE, in the hybrid system.

Hybridization process hopefully helps to get rid of falling in premature convergence and getting trapped in local optima. According to Liang et al.[13] the idea of this kind of hybridization gets more importance, when we are solving some complex constrained optimization problems where the ratio of the feasible region and search region is very small. It is found to be very tough to find the global solution of these problems.

Idea of the hybridization:

The purpose of incorporating these two giant techniques in a single one is to provide a better and more robust algorithm for solving constrained optimization problems. This kind of hybridization between DE and PSO could provide us a better solution for global optimization problems. Keeping in view the above observations and inspired by the concept of GDE, a tri-breakup-population based mechanism is proposed in this paper and applied in constrained engineering design problem. It initiates with a random population. Then find the fitness function of each individual in the population by using the bracket operator penalty. The strings are then sorted according to the increasing order of their function value. Now the population is being allowed to break in to three different groups A, B and C namely inferior group (first 1/3rd of the population), mid group (middle 1/3rd of the population) and superior group (last 1/3rd of the population), respectively. Of course the population size is kept fixed to a multiple of 3 to favor the tri-breakup mechanism. According to the local and global searching behavior of DE (as observed above), it is allowed to being employed in both the inferior and superior groups (i.e. to A and C). At the same time Particle Swarm Optimization (PSO) is used in the mid-group (i.e. to B) to overcome the shortcomings of DE. Therefore the synergy of DE-PSO-DE (DPD) is the hybrid method proposed in this paper.

4. Experimental Setup

Selection of Mutation operators for DEs in DPD

In DE Algorithm, the role of mutation operator is as a kernel operator that has the ability in the search of potential solutions. On the basis of

review of the past literature, researchers used many mutation strategies in the DE algorithm. For DE, selecting a mutation operator is indeed a difficult task. Therefore, in the present study, 8 different strategies of most efficient mutation operators have been reconsidered for analysis. The list of those strategies is reported in section II (B). While dealing with DPD, DE is being used two times in a generation. Hence, all combinations of mutation strategies are considered as:

DE (with 8 different strategies of mutation) + PSO + DE (with 8 different strategies of mutation) = DPD

Therefore a total of 8X8 = 64 combinations with all possible permutation of mutation operators, are generated by considering one case at a time and keeping PSO fixed. Some experiments are carried out so as to identify the top four combinations are (3, 3), (3, 7), (3, 5), (3, 1). Finally out of top four combinations check the performance which is defined in[14]. Top 4 combinations (i.e. all forms of DPD) are tested in two constrained engineering design problems reported in[15] include with different number of decision variables and a range of types (linear inequalities, nonlinear equalities, and nonlinear inequalities) and number of constraints. For this 100 independent runs with 10000 function evaluations are fixed to start the simulation with same parameter setting according to IV (B). Hence, to measure the winner combination, a Performance and its extended form is used and is defined as below:

$$\text{Performance} = \frac{1}{N_p}\sum_{i=1}^{N_p}\left(k_1 a_1^i + k_2 a_2^i + k_3 a_3^i\right),$$

subject to $k_1 + k_2 + k_3 = 1$ *and* $k_1 = k_2 = k_3$,

where

$$\alpha_1^i = \frac{Sr^i}{Tr^i}$$

$$\alpha_2^i = \begin{cases} \frac{Mo^i}{Ao^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0 \end{cases}$$

$$\alpha_3^i = \begin{cases} \frac{Mf^i}{Af^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0 \end{cases}$$

$$for \quad i = 1, 2, \ldots\ldots\ldots\ldots, N_p,$$

Sr^i = Number of successful runs of i^{th} problem

Tr^i = Total number of runs of i^{th} problem

Mo^i = Mean optimal objective function value obtained by an algorithm of i^{th} problem

Ao^i = Minimum of mean optimal objective function value obtained by all the algorithms of i^{th} problem

Mf^i = Average number of function evaluations of successful runs required by an algorithm in obtaining the solution of i^{th} problem

Af^i = Minimum of average number of function evaluations of successful runs required by all algorithms in obtaining the solution of i^{th} problem

N_p = Total number of problems.

Out of top 4 mutation combinations {(3, 3), (3, 7), (3, 5), (3, 1)} the performance of (3, 3) combination are better than other three shown in Fig.1. Hence, this combination i.e. (3, 3) is considered for further study.

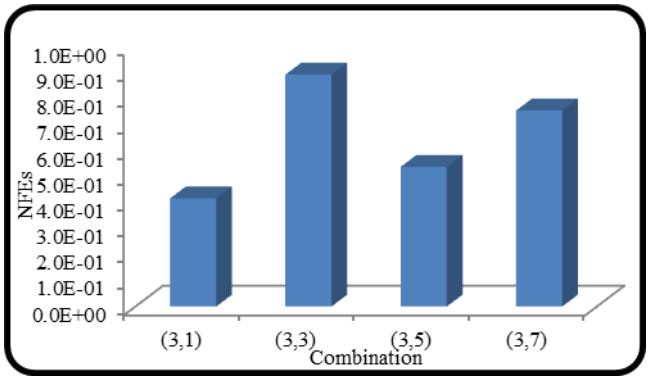

Fig.1. Performance evaluation for top 4 combinations (Under 2 engineering design problem)

Engineering Design Problem and Parameter Setting

A set of two well-known constrained engineering design problems[15] are carried out to evaluate the performance of DPD.

(1) E01: Welded Beam design problem

(2) E02: Pressure Vessel design problem

These problems have been previously solved using a variety of other techniques, which is useful to show the validity and effectiveness of the proposed algorithm. For each problem, 100 independent runs are performed and statistical results which give information about the robustness of the algorithm. Moreover, the statistical results provided by DPD were compared with some algorithms used in literature. For PSO, the acceleration constants C_1 and C_2 are both set to 2.0, the inertial weights (*w)* are set to 0.7298.

However for DE in DPD, the mutation factor in group 1 i.e. F_A=0.5 and for group 3 i.e. F_C=0.8; and the crossover weight CR_A=CR_C=0.9. The population size for DPD is set to a value, multiple of 3; it kept fixed to 51 for all problems to favour the population-tri-breakup system. As the problems studied in this work are minimization ones, the penalty function just adds a very high value *($R=1e^{10}$)* to the objective function. This penalty value was empirically chosen to be considerably bigger than the objective function values generated by the tested problems. Maximum numbers of function evaluation DPD are set as 10000. Values in "bold face" in tables represent the best obtained in the listed comparisons. The DPD are coded in C-Free Standard 4.0 and implemented on CORE i3, 2.8 GHz machine with 2GB of RAM, Environment.

5. Stimulation Results and Discussion for Two Engineering Design Problems

The best results and statistical results of the DPD on the two engineering design problems are summarized in Table (1-2) and Table 3 respectively. It includes the best, mean, worst and standard deviations over 100 independent runs. In Table 3, it shows that DPD can generate best solutions for all 100 runs. However, all problems deliver best value with low standard deviations. This illustrated that the results generated by DPD is robust. Values in "bold face" in tables represent the best obtained in the listed comparisons.

Table 1. Best solutions obtained for welded beam problem

Design variables	Methods				
	Lee and Geem[16]	Gandomi et al.[17]	Kazemzadeh Azad et al.[18]	SOPT	DPD
x_1	0.2442	0.2015	0.2054	0.20573	0.205724
x_2	6.2231	3.562	3.4783	3.47050	3.253253
x_3	8.2915	9.0414	9.0386	9.03663	9.036644
x_4	0.2443	0.2057	0.2057	0.20573	0.205729
$f(x)$	2.38	1.73121	1.72576	1.72485	1.695255
maxNFEs	110,000	50,000	20,000	10,000	10,000

Table 2. Best solutions obtained for pressure vessel problem

Design variables	Methods		
	Kazemzadeh Azad et al.[18]	SOPT	DPD
x_1	1.125	1.125	0.7781
x_2	0.625	0.625	0.3846
x_3	58.2895	58.2902	40.3196
x_4	43.6964	43.6927	200.0000
$f(x)$	7199.412	7199.359	**5884.689986**
maxNFEs	25,000	10,000	10,000

Table 3. Statistical results for Engineering design problems by different methods

Problems	Performance	Methods		
		K. Azad et al.[18]	SOPT	DPD
E01	Best	1.72576	1.72485	**1.695255**
	Average	1.773	1.72491	**1.695255**
	Worst	2.1376	1.72570	**1.695255**
	STD	0.0824	0.0001	**0.000E+00**
E02	Best	7199.412	7199.359	**5884.689986**
	Average	7347.105	7208.215	**5884.689986**
	Worst	9770.499	7342.977	**5884.689986**
	STD	420.07	29.16	**0.000E+00**

In order to show the effectiveness and superiority of DPD, it is compared with[16-18] and SOPT[15]. The results provided by compared approaches were directly taken from SOPT[15]. The best solution reported by DPD algorithm for welded beam design problem is f(X) = 1.695255 corresponding to decision variable X = [0.205724, 3.253253, 9.036644, 0.205729] and constraints [g_1(X), g_2(X), g_3(X), g_4(X), g_5(X), g_6(X), g_7(X)] = [-0.179754, -0.186979, -0.000005, -3.452407, -0.080724, -0.228310, -0.039577]. Also the best solution obtained by DPD for Pressure Vessel design problem f(X) = 5884.689986 corresponding to decision variable X = [0.7781, 0.3846, 40.3196, 200.0000] and constraints [g_1(X), g_2(X), g_3(X), g_4(X)] = [-2.95E-11, -7.15E-11, -1.35E-06, -40.0000]. Out of two engineering design problems DPD delivered best result compared to other algorithms in all manners.

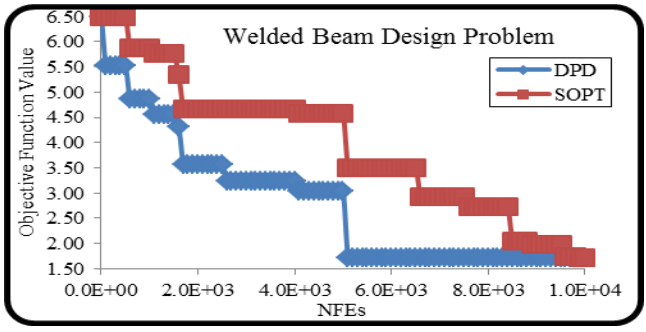

Fig. 2(a). Convergence for welded beam design problem

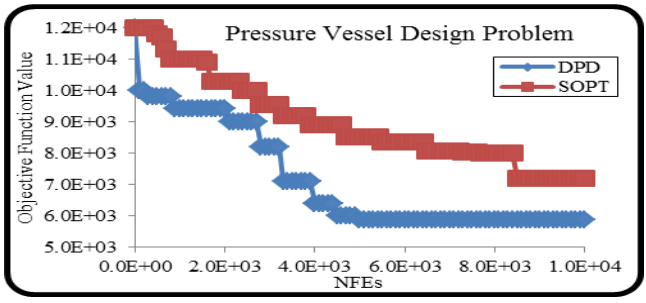

Fig. 2(b). Convergence for Pressure vessel design problem

As a conclusion, on the comparison above, DPD shows a very competitive performance with other algorithms in terms of the quality, the robustness, and the efficiency of search. Also, DPD is able to consistently find the best value with a very small standard deviation which indicates that the proposed DPD has a remarkable ability to solve constrained problems. Overall, among non-DPD[16-18] and SOPT[18] performs better, but it is worse than DPD. Therefore, SOPT is considered for convergence comparison graph with DPD in two engineering design problem. Staring from the same seed, SOPT and DPD algorithms allowed running over function evaluation for a fair comparison, it implies all the methods start from same initial population. For the above two engineering problems, the convergence graphs are shown in Fig 2(a-b). Undoubtedly from Fig 2(a-b), it is clear that for each of engineering problems DPD converges faster than SOPT. Additionally it can be observed that the DPD algorithm requires less computational effort *(according to NFEs)* than the other algorithms. Therefore, it is considered the fastest one with the least maximum number of function evaluation.

Conclusion and Future Works

This paper proposes a novel algorithm named DPD, which integrates PSO with DE. By using global information obtained from DE and PSO, the exploration and exploitation abilities of DPD algorithm are balanced. DPD uses an information exchange mechanism that helps to avoid the premature convergence and violating particles are reproduced from a memory in which some of the so far best design variables are saved. An investigated mutation strategy for DEs in DPD, enhance the local search ability and advance the convergence rate. The two strategies *Elitism* and *NRS* illustrate the attractiveness of the proposed method.

In order to demonstrate the effectiveness of the proposed method, it is applied to solve two well-known engineering design problems. The simulation results and comparisons provide evidence that the proposed DPD is superior in terms of various performance evaluation criteria such as mean, best, worst and standard deviation. The computation cost represented by the number of function evaluations of DPD is less than other existing techniques reported in literature. In general, it is very effective for solving constrained engineering optimization problems. Also, it is simple, robust, converges fast, and able to find the optimum solution in almost every run. Therefore, tri-beak-up technology for the population really makes the DE-PSO-DE faster and robust. Thus, DPD technique can be used as a good alternative for solving constrained engineering design problems. However, there are still some things to do in the future. Firstly, we will further validate DPD in the case of higher dimensions. Secondly, we also will take some

measures to improve the convergence speed during the evolutionary process. Additionally, different hybrid models of DE and PSO algorithm will be studied.

References

1. R. Storn and K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, **11(4)** (1997) 341-359.
2. J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. IEEE Int. Conf. Neural Networks*, **4** (1995) 1942-1948.
3. D. H. Wolpert and W. G. Macready, No Free Lunch Theorems for Optimization, Evolutionary Computation, *IEEE Transactions on Evolutionary Computing*, **1(1)** (1997) 67-82.
4. H. Liu, Z. X. Cai and Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, *Applied Soft Computing*, **10(2)** (2010) 629–640.
5. S. Khamsawang, P. Wannakarn and S. Jiriwibhakorn, Hybrid PSO-DE for solving the economic dispatch problem with generator constraints, *Proceedings of the IEEE International Conference on Computer and Automation Engineering*, **5** (2010) 135–139.
6. E. Nwankwor, A. Nagar and D. Reid, Hybrid differential evolution and particle swarm optimization for optimal well placement, *Computational Geosciences*, **17(2)** (2013) 1–20.
7. B. Xin, J. Chen, J. Zhang, H. Fang and Z. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and tax-onomy, *IEEE Transactions on Systems Man and Cybernetics Part C: Applications and Reviews*, **42(5)** (2012) 744–767.
8. A. Yadav and K. Deep, An efficient co-swarm particle swarm optimization for non-linear constrained optimization, *Journal of Computational Science*, http://dx.doi.org/10.1016/j.jocs.2013.05.011, (2013).
9. Z. Michalewicz, A survey of constraint handling techniques in evolutionary computation methods, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, MIT Press, Cambridge, (1995) 135–155.
10. K. Deb, *Optimization for Engineering Design: Algorithms and Examples*, Prentice-Hall of India, New Delhi, 1995.
11. H. Zhang and M. Ishikawa, An extended hybrid genetic algorithm for exploring a large search space, *2nd International conference on autonomous robots and agents*, (2004) 244–248.

12. M. F. Han, S. H. Liao, J. Y. Chang and C. T. Lin, Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems, *Applied Intelligence*, **39(1)** (2013) 41-56.

13. J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello and K. Deb, Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization, *Journal of Applied Mechanics*, **41** (2006).

14. K. Deep and K. N. Das, Performance improvement of real coded genetic algorithm with Quadratic Approximation based hybridization, *Int. J. Intelligent Defence Support Systems*, **2(4)** (2009) 319-334.

15. O. Hasançebi and S. K. Azad, An efficient metaheuristic algorithm for engineering optimization: SOPT, *Int. J. Optim. Civil Eng.*, **2(4)** (2012) 479-487.

16. K. S. Lee and Z. W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, *Comput Meth Appl Mech Eng*, **194** (2005) 3902–3933.

17. A. H. Gandomi, X. S. Yang and A. H. Alavi, Mixed variable structural optimization using firefly algorithm, *Comput Struct*., **89** (2011) 2325–2336.

18. S. Kazemzadeh Azad, O. Hasançebi and O. K. Erol, Evaluating efficiency of Big- Bang Big-Crunch algorithm in benchmark engineering optimization problems, *Int. J. Optim Civil Eng*., **1(3)** (2011) 495-05.