International Journal on Artificial Intelligence Tools
Vol. 21, No. 3 (2012) 1240013 (25 pages)

DOI: 10.1142/S0218213012400131

World Scientific
www.worldscientific.com

SWARM DIRECTIONS EMBEDDED DIFFERENTIAL EVOLUTION FOR FASTER CONVERGENCE OF GLOBAL OPTIMIZATION PROBLEMS

MUSRRAT ALI
Department of Computer Science, Sungkyunkwan University
Suwon 440-746, Republic of Korea
musrrat.iitr@gmail.com

MILLIE PANT
Department of Paper Technology, Indian Institute of Technology Roorkee
Roorkee, Uttarakhand 247667, India
millifpt@iitr.ernet.in

AJITH ABRAHAM
Center of Excellence for Quantifiable Quality of Service
Norwegian University of Science and Technology, Norway
ajith.abraham@ieee.org

CHANG WOOK AHN
Department of Computer Science, Sungkyunkwan University
Suwon 440-746, Republic of Korea
cwan@skku.edu

In the present study we propose a new hybrid version of Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms called Hybrid DE or HDE for solving continuous global optimization problems. In the proposed HDE algorithm, information sharing mechanism of PSO is embedded in the contracted search space obtained by the basic DE algorithm. This is done to maintain a balance between the two antagonist factors; exploration and exploitation thereby obtaining a faster convergence. The embedding of swarm directions to the basic DE algorithm is done with the help of a "*switchover constant*" called α which keeps a record of the contraction of search space. The proposed HDE algorithm is tested on a set of 10 unconstrained benchmark problems and four constrained real life, mechanical design problems. Empirical studies show that the proposed scheme helps in improving the convergence rate of the basic DE algorithm without compromising with the quality of solution.

Keywords: Differential evolution; particle swarm optimization; hybridization; global optimization.

1. Introduction

Optimization problems arise in various disciplines such as engineering designs, agricultural sciences, manufacturing systems, economics, physical sciences, pattern recognition etc. in fact optimization techniques are being extensively used in various

spheres of human activities, where decisions have to be taken in some complex situation which can be represented by mathematical models. Optimization can thus be viewed as a kind of decision making, or more specifically, as one of the major quantitative tools in network of decision making, in which decisions have to be taken to optimize one or more objectives under some prescribed set of circumstances. Significant amount of work has been done in the field optimization which mainly revolves around developing efficient methods for solving mathematical models of optimization problems. Some of the classical optimization techniques include Methods Linear Programming (LP) and traditional nonlinear (NL) techniques like Bellman's principle, Lagrange's multiplier method, quadratic programming approach etc. A detailed study of classical optimization methods can be found in Refs. 28, 8 and 43. While LP approaches are restricted to linear models, the problem with most of the traditional NL techniques is that they are gradient based and hence cannot be applied for finding out the solution to the problems which have a rough or discontinuous surface. Consequently, researchers working in the field of optimization have laid much attention on a class of derivative free techniques that can be applied for solving complex optimization models.

One class of derivative free techniques consists of nature inspired evolutionary search algorithms. These techniques have gained a lot of popularity in the recent years because of their ability to deal with complex optimization problems which are otherwise difficult to solve using traditional methods. Some well known evolutionary/nature inspired algorithms are Genetic Algorithms,[25] Evolutionary Programming,[24] Evolutionary Strategies[7] Particle Swarm optimization[43] and Differential Evolution[51] etc. The working of all these algorithms is based on some biological or social metaphor.

Out of these, PSO and DE are comparatively newer than others. These are stochastic, population based search techniques and have been applied successfully to a wide range of problems occurring in various disciplines.[2–4,10,16–18,22,23,26,35,36,41,44,45,52,54,55,58,60] A number of variants of these algorithms have been developed in the past decade to improve their performance.[5,9,29,42,46,49,50,57]

One class of modification consists of integrating DE and PSO algorithms where the advantages of both the algorithms are blended together to form a new algorithm. DE has the advantage of not being biased towards any prior defined distribution for sampling mutational step sizes and its selection operator follows a hill-climbing process. According to the literature, the DE algorithm performs very well in the initial stages however as the search proceeds, the convergence rate of DE slows down gradually.

On the basis of practical experience, researchers have shown that DE may sometimes stop proceeding toward the global optimum even though the population has not converged to a local optimum or any other point.[34] There is a possibility that the new points may enter the population but the algorithm does not show any progress in finding any better solutions. This situation is termed as *stagnation*. DE also suffers from the problem of premature convergence, where the population converges to some local optima of a multimodal objective function, losing its diversity. The probability of stagnation depends on how many different potential trial solutions are available and also on their capability to enter into the population of the subsequent generations.[34]

Further, like other evolutionary computing algorithms, the performance of DE deteriorates with the growth of the dimensionality of the search space as well. A good volume of work is devoted in the direction of improvement of DE algorithm. These modifications may be differentiated as variation in the probabilities of crossover and mutation, development of new mutation or crossover operators, development of adaptive parameters[1,5,9,12,19,29,34,37,39,40,42,46,57,59] etc. Hybridization of DE with some local search technique or with some stochastic technique can be found in Refs. 11, 14, 27, 30–32, 38, 53, 56, 61 and 62.

The idea behind hybridization is to merge the positive features of two (or more than two) algorithms in order to obtain a new algorithm which is better than the two parent algorithms which are used for hybridization. In the present study we propose a hybrid version of DE and PSO called hybridized DE or HDE. The proposed HDE algorithm works in two phases; the DE phase (or the first phase) and the PSO phase (or the second phase). The change of phases is decided according to a *switchover constant* α. By embedding the information sharing mechanism of PSO in the later stages of the search we try to push the particles towards a local attractor which is hopefully near the global optimum.

The remaining paper is organized as follows: In Sec. 2, we give a brief overview of the hybridized DE-PSO algorithms. In Sec. 3, basic DE and PSO algorithms are described. The proposed HDE algorithm is given in Sec. 4. Benchmark problems and experimental settings are given in Sec. 5 and numerical results are given in Sec. 6. Finally, the conclusions based on the present study are drawn in Sec. 7.

2. A Brief Literature Review on Earlier Work Done

As mentioned in the previous section, both DE and PSO have undergone a plethora of changes since their development in 1995. However, not much instances are available in literature which combines the features of DE and PSO together in a single algorithm. In this section we give a brief review of some earlier work in which the two algorithms are merged together.

Hendtlass[47] used the DE perturbation approach to adapt particle positions. In his algorithm, named SDEA, particles' positions are updated only if their offspring have better fitness. The DE reproduction process is applied to the particles in swarm at specified intervals. At the specified intervals, the PSO swarm serves as the population for DE algorithm, and the DE is executed for a number of generations. After execution of DE, the evolved population is further optimized using PSO. Hendtlass applied his algorithm SDEA on four unconstrained benchmark problems for different dimensions and for different population sizes.

Zhang and Xie[62] used different techniques in random, rather than combining them, in their Differential Evolution (DE) PSO (DEPSO). In this case the DE and canonical PSO operators were used on alternate generations; when DE was in use, the trial mutation replaced the individual best at a rate controlled by a crossover constant and a random dimension selector that ensured at least one mutation occurred each time.

Kannan *et al.*[20] applied DE to each particle for a finite number of iterations, and replaced the particle with the best individual obtained from the DE process.

Talbi and Batauche[53] proposed an algorithm named DEPSO. It differs from the DEPSO algorithm proposed by Zhang and Xie as the DE operators are applied only to the best particle obtained by PSO. They applied their algorithm on medical image processing problem.

In Hao *et al.*'s[63] hybrid version, which is also named as DEPSO, the candidate solution is generated either by DE or by PSO according to some fixed probability distribution. They applied their algorithm for solving unconstrained global optimization problems.

Das *et al.*[18] proposed a scheme of adjusting velocities of the particles in PSO with a vector differential operator borrowed from the DE family. In their proposed PSO-DV algorithm, they omitted the cognitive term of the canonical PSO and updated the particle velocities by a new term containing the weighted difference (inspired by DE mutation scheme) of the position vectors of any two distinct particles randomly chosen from the swarm. They also applied their algorithm on a test suite of selected unconstrained benchmark problems.

Omran *et al.*[38] proposed a hybrid version of Bare Bones PSO and DE called it BBDE. In their approach, they combined the concept of barebones PSO with self adaptive DE strategies. The mutation operator of DE is used to explore around the current attractor by adding a difference vector to the attractor. Crossover is done with randomly selected personal best as these personal bests represent a memory of the best solution found by individuals since the start of the search process. They validated their algorithm on a set of unconstrained benchmark problems and also applied to image classification problem.

The work of Jose *et al.*[64] evaluated a Particle Swarm Optimizer hybridized with Differential Evolution and applied it to the Black-Box Optimization Benchmarking for noisy functions. Their algorithm was once named as DEPSO algorithm. In this version of DEPSO, the differential variation schemes of DE are used for updating the velocities of the swarm particles.

Zhang *et al.*[61] developed a hybrid of DE and PSO called DE-PSO, where three alternative updating strategies are used. The DE updating strategy is executed once in every l generations and if a particle's best encountered position and the position of its current best neighbor are equal then random updating strategy is executed otherwise PSO updating strategy is used.

Liu *et al.*[65] proposed a novel hybrid algorithm named PSO-DE, in which 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.

Copanio *et al.*[66] developed a Superfit Memetic Differential Evolution (SFMDE) by hybridizing DE and PSO with two other local search methods; Nelder Mead algorithm and Rosenbrock algorithm. In SFMDE, PSO assists DE in the beginning of the optimization process to generate a "*super-fit individual*", the local searches are then applied adaptively by means of a parameter which measures the quality of super-fit

Table 1. A summary of hybridized PSO + DE algorithms.

Author/s	Year	Name of the Algorithm	Application/s
Hentdlass[47]	2001	SDEA	Unconstrained global optimization
Zhang and Xie[62]	2003	DEPSO	Unconstrained global optimization
Kannan *et al.*[20]	2004		Generation Expansion planning
Talabi and Batouche[53]	2004	DEPSO	Medical Image Processing
Hao *et al.*[63]	2007	DEPSO	Unconstrained global optimization
Das *et al.*[18]	2008	PSO-DV	Engineering Design
Omran *et al.*[38]	2008	BBDE	Unconstrained Optimization Problems and Image Processing
Jose *et al.*[64]	2009	DEPSO	Noisy functions
Zhang *et al.*[61]	2009	DE-PSO	Unconstrained Optimization
Liu *et al.*[65]	2009	PSO-DE	Constrained optimization and engineering problems
Caponio *et al.*[66]	2009	SFMDE	Unconstrained global optimization and engineering design problems
Xu and Gu[67]	2009	PSOPDE	Unconstrained global optimization
Pant *et al.*[68]	2009	DE-PSO	Unconstrained global optimization
Khamsawang *et al.*[69]	2010	PSO-DE	Power Systems

individual. SFMDE was applied for solving unconstrained standard benchmark problems and two engineering problems.

Xu and Gu[67] proposed Particle Swarm Optimization with prior crossover differential evolution (PSOPDE). Their version is distinct from other PSO-DE hybrids in three ways; (1) the particles in the swarm are not just led towards global and personal best positions but also depend on the average position and velocity of the particles (2) DE is integrated with PSO for local search and (3) a new crossover operation between the target and an extra population is implemented before the DE component. They applied their algorithm for solving five unconstrained benchmark problems.

Khamsawang *et al.*[69] 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. In PSO-DE, the mutation operators of the differential evolution are used for improving diversity exploration of PSO and are activated if velocity values of PSO are near to zero or violate the boundary conditions.

The DE-PSO algorithm suggested by Pant *et al.*[68] starts like the usual DE algorithm. it enters the PSO phase if the optimality criteria are not met by the DE algorithm. They tested their algorithm on a set of unconstrained benchmark problems. A brief summary of hybrid PSO and DE algorithms is given in Table 1.

3. Differential Evolution (DE) and Particle Swarm Optimization (PSO)

DE was proposed by Storn and Price[51] in 1995. It soon became a popular tool for solving global optimization problems because of several attractive features like having fewer

control parameters, ease in programming, efficiency etc. DE is similar to GAs in the sense that it uses same evolutionary operators like mutation, crossover and selection for guiding the population towards the optimum solution. Nevertheless, it's the application of these operators that makes DE different from GA. The main difference between GAs and DE is that; in GAs, mutation is the result of small perturbations to the genes of an individual while in DE mutation is the result of arithmetic combinations of individuals. Also in DE, mutation plays a prominent role whereas, in GA, crossover is the major operator. At the beginning of the evolution process, the mutation operator of DE favors exploration. As evolution progresses, the mutation operator favors exploitation. Hence, DE automatically adapts the mutation increments (i.e. search step) to the best value based on the stage of the evolutionary process. Mutation in DE is therefore not based on a predefined probability density function. Moreover, GA may work even without the presence of a mutation operator as crossover is the prime operator of GA but the main component of DE is the generation of mutant vector. Also in DE mutation is applied before crossover whereas in GA, mutation is always applied after crossover. Throughout the study we shall consider the mutation strategy DE/rand/1/bin also known as the classical version of DE. The three main equations used in DE are given as follows:

$$U_{i,G+1} = X_{r3,G} + F*(X_{r1,G} - X_{r2,G}) \tag{1}$$

$$t_{j,i.G+1} = \begin{cases} u_{j,i.G+1} & \text{if } r \text{ and } j \le C_r \vee j = k \\ x_{j,i.G} & \text{otherwise} \end{cases} \tag{2}$$

$$X_{i.G+1} = \begin{cases} T_{i.G+1} & \text{if } f(T_{i.G+1}) \le f(X_{i.G}) \\ X_{i.G} & \text{otherwise} \end{cases} \tag{3}$$

Equations (1), (2) and (3) represent the mutation, crossover and selection operations respectively, for DE. F is the scaling factor and Cr is the probability of crossover.

PSO is a multi-agent parallel search technique developed by Kennedy and Eberhart in 1995,[33] inspired by social behavior of bird flocking or fish schooling. The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors (or colleagues). Each particle maintains a memory of its previous best position and also the best position among all the particles. During each generation each particle is accelerated toward the particles previous best position and the global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a number of times or until a minimum error is achieved.

$$V_i^{G+1} = w^G V_i^G + c_1 r_1 (P_i^G - X_i^G) + c_2 r_2 (P_{\text{best}}^G - X_i^G) \tag{4}$$

$$X_i^{G+1} = X_i^G + V_i^{G+1} \tag{5}$$

$$w^G = (w_{\max} - w_{\min}) \times \frac{(\text{maxiteration-Iteration})}{\text{Iteration}} + w_{\min} \tag{6}$$

Equations (4) and (5) represent the velocity and position equations of the swarm particles. Equation (6) gives the inertia weight taken for the present study. P_i^G represents the personal best position of the particle and P_{best}^G represents the global best particle. c_1, c_2 represents the acceleration constants and r_1, r_2 are uniformly distributed random numbers in the range (0, 1).

4. Proposed Hybrid Differential Evolution (HDE)

In hybridized algorithms it is very crucial to decide how the algorithms should be merged in order to get the maximum benefit. In the proposed HDE algorithm, the initial search space is contracted using the DE algorithm and then PSO is applied to further refine the search space with the hope of finding the global optimal solution. The change of DE phase to PSO phase is decided with the help of a switchover constant denoted by α. The final contracted population obtained by DE is arranged in ascending order of fitness function values and upper half of the population undergoes the PSO operation i.e. to say that PSO is applied on only half of the population consisting of elite solutions. The working of HDE can be explained with the help of the following steps.

Step 1. Generate randomly NP vectors, each of n dimensions:
$X_{i,j} = X_{\min,j} + \text{rand}(0, 1)(X_{\max,j} - X_{\min,j})$, where $X_{\min,j}$ and $X_{\max,j}$ are lower and upper bounds for jth component respectively, rand(0, 1) is a uniformly distributed random number between 0 and 1.

Step 2. Calculate the objective function value $f(X_i)$ for all X_i.

Step 3. Corresponding to target vector X_i select three distinct points from population and generate perturbed individual U_i using equation (1).

Step 4. Recombine the each target vector X_i with perturbed individual generated in step 3 to generate a trial vector T_i using equation (2).

Step 5. Check whether each variable of the trial vector is within range. If not keep it within range using $T_{i,j} = 2 * X_{\min,j} - T_{i,j}$, if $T_{i,j} < X_{\min,j}$ and $T_{i,j} = 2 * X_{\max,j} - T_{i,j}$, if $T_{i,j} > X_{\max,j}$, otherwise go to Step 6.

Step 6. Calculate the objective function value for vector T_i.

Step 7. Choose better of the two (function value at target and trial point) using equation (3) for next generation.

Step 8. Check whether $|f_{\max} - f_{\min}| < \alpha$ (given threshold) value. If yes then switch-over to PSO (Step 10); otherwise go to Step 9.

Step 9. Check whether maximum number of function evaluation has been completed. If yes, stop; otherwise go to Step 3.

Step 10. Sort the final population obtained by DE algorithm in ascending order. Take first NP/2 individuals as initial swarm; assign initial velocity zero, personal best position same as particle position and global best position particle position having minimum function value.

Step 11. Update velocity V_i^{G+1} and position X_i^{G+1}, according to equations (4) and (5) of all NP/2 particles.
Step 12. Evaluate objective function value $f(X_i^{G+1})$ for all X_i^{G+1}.
Step 13. Update particle best position: if $f(X_i^{G+1}) < f(P_i)$ then $f(P_i) = f(X_i^{G+1})$ and $P_i = X_i^{G+1}$.
Step 14. Find global best P_{best} position of particle and update inertia weight.
Step 15. Check whether convergence criterion is met. If yes, stop: otherwise go to Step 16.
Step 16. Check whether maximum number of function evaluation has been completed. If yes, then stop; otherwise go to Step 11.

The parameter α defined in Step 8, plays an important role in HDE. It keeps a track of the contraction of the search space in order to initiate the PSO algorithm.

5. Benchmark Problems and Experimental Settings

5.1. *Benchmark and real life problems*

To analyze the performance of the proposed HDE algorithm we tested it on 10 unconstrained standard bench mark problems taken from literature. All these problems are scalable in nature i.e. their dimension can be varied in order to increase their complexity. We have tested these problems for dimensions 10 and 30. The mathematical model of the problems along with the true global minimum is given the Appendix A.

To further analyze the efficiency of HDE we used it for solving four real life mechanical design problems. All the four problems are constrained in nature. The constraints are dealt with by using the tournament based approach proposed by Deb[21] for solving constrained optimization problems. According to this rule:

- Between two feasible solutions, the one with the highest fitness value wins.
- If one solution is feasible and the other one is infeasible, the feasible solution wins.
- If both solutions are infeasible, the one with the lowest sum of constraint violation is preferred.

This method does not require a penalty factor as the selection procedure only performs pair wise comparisons. The feasible solutions therefore have fitness equal to their objective function value, and the use of constraint violation in the comparisons aims to push the infeasible solutions towards the feasible region.

5.1.1. *Experimental settings*

As discussed in Sec. 4, HDE starts with the usual DE algorithm for which, the lower limit for population size, NP, is 4 since the mutation process requires at least three other chromosomes for each parent. While testing the algorithms, we began by using the optimized control settings of DE. Population size, NP can always be increased to help maintain population diversity. As a general rule, an effective NP is between $3 * n$ and

$10 * n$, but can often be modified depending on the complexity of the problem. For the present study we performed several experiments with the population size as well as with the crossover rate and mutation probability rate and observed that for problems up to dimension 30 a population size of $10 * n$ is sufficient. Values of scale F, outside the range of 0.4 to 1.2 are rarely effective and $F = 0.5$ can be considered to be a good initial choice. In general higher value of C_r help in speeding up the convergence rate therefore in the present study we have taken $C_r = 0.5$.

HDE also contains an additional parameter α, which keeps a track of the contraction of search space and decides the initiation of PSO algorithm. We observed the preformance of the proposed HDE algorithm for three values of α; 0.5, 0.05 and 0.005.

The PSO algorithm in HDE is applied to a region which has already contrcted i.e. to say the PSO algorithm is applied to a small search space (hopefully) near the global minima. Therefore, we have taken the inertia weight w which decrease uniformly from 0.4 to 0.2. Also in order to give more weight to the global best position, the acceleration coeffient C_1 and C_2 are taken as 0.5 and 2.0 respectively. All the algorithms are executed on a PIV PC, using DEV C++, thirty times for each problem. Random numbers are generated using the inbuilt random number generator *rand* ( ) *function* available in DEVC++.

In every case, a run was terminated when the best function value obtained is less than a threshold (10^{-4}) for the given function or when the maximum number of function evaluation (NFE = 10^6) was reached. In order to have a fair comparison, these settings are kept the same for all algorithms over all benchmark functions during the simulations.

6. Numerical Results

6.1. *Numerical results for unconstrained benchmark problems*

The performance of the proposed HDE algorithm, for different values of α, is analyzed on a set of ten unconstrained benchmark problems and the numerical results are compared with the traditional DE algorithm. The performance measures used for comparison include average fitness function value, standard deviation (Std), average numbers of functions evaluations and average CPU time taken. The corresponding results are given in Tables 2–5.

Under the given parameter settings, it can be seen that the traditional DE was not able to achieve the desired accuracy of 10^{-4} in case of f_{RB} (Rosenbrock function) whereas, for f_{GW} (Griewanks function) HDE was not able to achieve the desired accuracy for $\alpha = 0.5$. For the remaining values of α (= 0.05 and 0.005) HDE was able to achieve the desired accuracy for all the test cases.

From the numerical results it can be seen that although in terms of average fitness function value, the proposed HDE algorithm does not show much improvement in comparison to the basic DE, but in terms of NFE and CPU time taken, the proposed HDE algorithm performs much better than the basic DE algorithm. If discuss the sensitivity of

Table 2. DE versus HDE (for α = 0.5, 0.05, 0.005) in terms of average fitness function value and standard deviation (Std) for unconstrained benchmark problems of dimension 10.

Fun	DE	HDE (α = 0.5)	HDE (α = 0.05)	HDE (α = 0.005)
	Fitness (Std)	Fitness (Std)	Fitness (Std)	Fitness (Std)
f_{SP}	1.40711e-05 (5.17314e-06)	**5.62630e-06** (3.91721e-06)	1.22108e-05 (7.65373e-06)	8.8269e-06 (5.16438e-06)
f_{ACK}	5.94765e-05 (1.0848e-05)	3.56962e-05 (1.30165e-05)	5.44005e-05 (2.94382e-05)	**3.43113e-05** (1.11886e-05)
f_{RG}	1.13561e-05 (3.38399e-06)	9.00661e-06 (5.72322e-06)	1.16375e-05 (7.7549e-06)	**6.83095e-06** (5.40267e-06)
f_{RB}	–	**9.02526e-14** (0.00340)	0.00096 (0.00205)	0.00063 (0.00064)
f_{GW}	1.46893e-05 (3.77339e-06)	–	2.74695e-06 (8.24084e-06)	**1.03926e-05** (5.69282e-06)
f_{LM2}	1.60214e-05 (4.07044e-06)	9.83868e-006 (8.48897e-06)	**7.96413e-06** (5.48033e-06)	8.12357e-06 (5.08711e-06)
f_{pp}	–45.77850 (4.25827e-06)	–45.77850 (4.24994e-06)	–45.77850 (6.03048e-06)	–45.77850 (2.57806e-06)
f_{sis}	–3.49998 (4.62082e-06)	–3.49999 (9.00903e-06)	–3.49999 (7.38833e-06)	–3.49999 (4.48909e-06)
f_{st}	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)
f_{SWF}	0.00024 (4.14499e-06)	0.00023 (4.76434e-06)	0.00023 (1.35658e-05)	0.00023 (5.46631e-06)

Table 3. DE versus HDE (for α = 0.5, 0.05, 0.005) in terms of average fitness function value and standard deviation (Std.) for unconstrained benchmark problems of dimension 30.

Fun	DE	HDE (α = 0.5)	HDE (α = 0.05)	HDE (α = 0.005)
	Fitness (Std)	Fitness (Std)	Fitness (Std)	Fitness (Std)
f_{SP}	4.96717e-05 7.77918e-06	**2.38679e-05** 3.29303e-05	9.01265e-05 4.05859e-05	6.02874e-05 1.67625e-05
f_{ACK}	**0.00014** 1.66159e-05	0.00047 0.00061	0.00024 0.00011	0.00018 8.38164e-05
f_{RG}	105.858 6.16943	104.131 6.20312	**87.8904** 6.01522	89.1484 5.22234
f_{RB}	15.1913 1.70576	**2.65097e-15** 1.55107e-10	13.6827 1.00897	13.8989 1.63056
f_{GW}	5.07342e-05 5.83072e-06	0.00744 0.00084	**4.43105e-05** 4.57603e-05	5.31735e-05 1.42185e-05
f_{LM2}	**4.73725e-05** 9.83696e-06	0.00031 0.00041	6.89487e-05 2.48868e-05	5.45789e-05 2.62944e-05
f_{pp}	–997867.0 2.45355e-06	–997867.0 3.341784e-05	–997867.0 3.22517e-05	–997867.0 1.31195e-05
f_{sis}	–3.49995 6.58779e-06	–2.33863 .00785	–2.45923 0.00067	–3.49985 3.20262e-05
f_{st}	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0
f_{SWF}	0.00072 7.56772e-06	0.00081 9.5201e-05	0.00075 6.24289e-05	0.00073 2.61436e-05

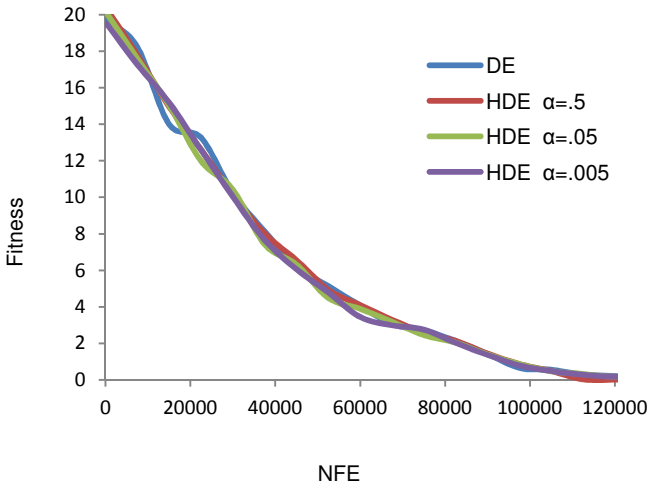
Table 4. DE versus HDE (for α = 0.5, 0.05, 0.005) in terms of average number of functions evaluations (NFE) and average CPU time taken (in seconds) to obtain an accuracy of 10^{-4} for unconstrained benchmark problems of dimension 10.

Fun	DE		HDE (α = 0.5)		HDE (α = 0.05)		HDE (α = 0.005)	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time
f_{SP}	14300	0.30	**8415**	0.20	8835	0.20	10740	0.20
f_{ACK}	26680	0.60	**13690**	0.30	16595	0.40	19885	0.50
f_{RG}	96400	2.20	**89345**	2.10	92090	2.20	95120	2.20
f_{RB}	10^6	–	10^6	–	208190	10.60	316155	17.90
f_{GW}	94690	2.40	10^6	–	88400	2.30	87290	2.20
f_{LM2}	17540	0.90	**10172**	0.44	13185	0.70	14325	0.70
f_{pp}	14290	0.70	**7760**	0.40	9400	0.40	11545	0.60
f_{sis}	30940	0.20	31300	0.20	**24395**	0.20	27695	0.20
f_{st}	10980	0.50	**11030**	0.30	11300	0.30	10690	0.30
f_{SWF}	25850	0.20	**19480**	0.10	21440	0.20	22570	0.10

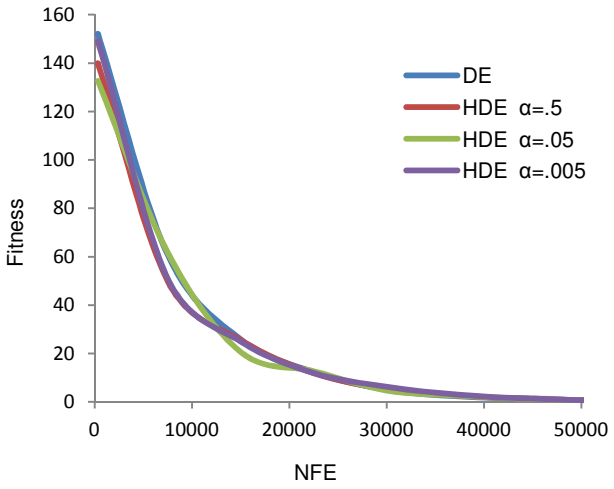
Table 5. DE versus HDE (for α = 0.5, 0.05, 0.005) in terms of average number of functions evaluations (NFE) taken to obtain accuracy of 10^{-4} for unconstrained benchmark problems of dimension 30.

Fun	NFE			
	DE	HDE (α = 0.5)	HDE (α = 0.05)	HDE (α = 0.005)
f_{SP}	146400	**90195**	90270	111465
f_{ACK}	256710	**122610**	152310	187530
f_{RG}	10^6	10^6	10^6	10^6
f_{RB}	10^6	10^6	10^6	10^6
f_{GW}	222540	**90150**	140583	186255
f_{LM2}	182970	**104950**	131325	149370
f_{pp}	225900	**149550**	171150	190350
f_{sis}	549675	443250	421650	429000
f_{st}	126060	125610	**121950**	125580
f_{SWF}	369450	**283395**	305325	324120

the parameter α, then once again from the average fitness function value we cannot conclude anything concrete because the numerical results are more or less similar for all values of α, which are similar to the basic DE results, which in turn are in the vicinity of the true minima. However, from Table 3, we can see that when the value of α is 0.5, we get the best results in terms of NFE and CPU time with an average improvement of around 23% in case of NFE and CPU time. However, for other values of α also the results are better than the basic DE algorithm. Performance graphs of few selected functions for different values of α are given in Fig. 1(a) and 1(b).



(a)



(b)

Fig. 1. Performance graph of (a) Ackley's function and (b) Sphere function.

6.2. *Numerical results for constrained real life problems*

In case of real life problems also, the proposed HDE performed better than the basic DE algorithm and the results available in literature,[6,13,15,20,48] for all the four test cases. Once again the superior performance of HDE algorithm is more evident from the NFE it takes to reach the optimum solution. In terms of other performance measures also like average fitness function value and average CPU time, the proposed HDE outperformed the other algorithms. The corresponding results of real life problems are given in Tables 6 and 7. Performance curves of real life problems are given in Figs. 2–5.

Table 6. A comparison of numerical results of PVD and T/CSD problems using DE, HDE (α = 0.05) and other algorithms available in literature in terms of Average Fitness function value, Number of function evaluations (NFE) Standard Deviation (std) and Time (in sec).

Pressure vessel design (PVD)						
Item	DE	HDE (α = 0.05)	SiC-PSO[13]	iGSO[48]	Coello[15]	GeneAS[20]
x_1	0.778169	0.778169	0.812500	0.812500	0.812500	0.9375
x_2	0.384649	0.384649	0.437500	0.437500	0.437500	0.5000
x_3	40.3196	40.3197	42.098445	42.098446	40.3239	48.3290
x_4	200	200	176.636595	176.636596	200	112.6790
$g_1(X)$	–5.59078e-10	–3.73187e-010	–4.500e-15	–3.40e-10	–0.034324	–0.004750
$g_2(X)$	–3.0915e-009	–7.13005e-011	–0.035880	–0.035881	–0.052847	–0.038941
$g_3(X)$	–0.0022058	–7.34673e-005	–1.164e-10	–0.000029	–27.105845	–3652.8738
$g_4(X)$	–40	–40.0005	–63.363404	–63.363404	–40.0000	–127.321000
$f(X)$	5885.3301	**5885.3300**	6059.714335	6059.71400	6288.7445	6410.3811
NFE	40740	**31915**	NA	26000	NA	NA
Std	1.25551e-05	2.57304e-4	NA	NA	NA	NA
Time (*sec*)	0.5	**0.3**	NA	12.34	NA	NA
Spring design (T/CSD)						
Item	DE	HDE (α = 0.05)	SiC-PSO[13]	iGSO[48]	Coello[15]	Arora[6]
x_1	0.0514312	0.0518374	0.051583	0.051691	0.051480	0.053396
x_2	0.350492	0.360296	0.354190	0.356765	0.351661	0.399180
x_3	11.6701	11.0822	11.438675	11.286172	11.632201	9.185400
$g_1(X)$	–0.00038971	–1.11544e-007	–2.000e-16	–2.0950e-11	–0.002080	0.000019
$g_2(X)$	–0.00012457	–3.36314e-010	–1.000e-16	–1.3020e-11	–0.000110	–0.000018
$g_3(X)$	–4.03868	–4.06079	–4.048765	–4.053880	–4.026318	–4.123842
$g_4(X)$	–0.732051	–0.725244	–0.729483	–0.727696	–0.731239	–0.698283
$f(X)$	0.0126737	**0.0126656**	0.012665	0.012665	0.0127048	0.127302737
NFE	9370	**4945**	NA	8000	NA	NA
Std	4.05487e-005	1.77850e-04	NA	NA	NA	NA
Time (*sec*)	0.2	**0.1**	NA	8.17	NA	NA

Table 7. A comparison of numerical results of SRD and WBD problems using DE, HDE (α = 0.05) and other algorithms available in literature in terms of Average Fitness function value, Number of function evaluations (NFE) Standard Deviation (std) and Time (in sec).

Speed Reducer (SRD)			
Item	DE	HDE (α = 0.05)	SiC-PSO[13]
x_1	3.5	3.5	3.500000
x_2	0.7	0.7	0.700000
x_3	17	17	17.00000
x_4	7.3	7.30012	7.30000
x_5	7.8	7.8	7.80000
x_6	3.35021	3.35021	3.350214
x_7	5.28668	5.28668	5.286683
$g_1(X)$	–0.0739153	–0.0739153	–0.073915
$g_2(X)$	–0.197999	–0.197999	–0.197998
$g_3(X)$	–0.499172	–0.499148	–0.499172
$g_4(X)$	–0.901472	–0.901472	–0.901471
$g_5(X)$	–2.381e-008	–1.64302e-009	0.000000
$g_6(X)$	–1.66315e-009	–2.97741e-010	–5.000e-16
$g_7(X)$	–0.7025	–0.7025	–0.702500
$g_8(X)$	–3.07603e-010	–4.98685e-011	–1.000e-16
$g_9(X)$	–0.583333	–0.583333	–0.583333
$g_{10}(X)$	–0.0513257	–0.0513413	0.051325
$g_{11}(X)$	–0.0108524	–0.0108524	–0.010852
$f(X)$	2996.35	**2996.347101**	2996.348165
NFE	8925	**7867.5**	NA
Std	9.02658e-06	**1.35461e-03**	NA
Time (*sec*)	0.4	**0.3**	NA

Table 7. (*Continued*)

Welded Beam Design(WBD)			
Item	DE	HDE ($\alpha = 0.05$)	SiC-PSO[13]
x_1	0.205727	0.182634	0.205729
x_2	3.47055	3.81039	3.470488
x_3	9.03667	9.58501	9.036624
x_4	0.20573	0.182863	0.205729
$g_1(X)$	–0.0681061	–0.00693705	–1.819e-12
$g_2(X)$	–0.458633	–0.14485	–0.003721
$g_3(X)$	–3.57774e-06	–0.000228855	0.00000
$g_4(X)$	–3.43296	–3.49465	–3.432983
$g_5(X)$	–0.0807269	–0.0576342	–0.080729
$g_6(X)$	–0.235541	–0.236368	–0.235540
$g_7(X)$	–2055.41	–0.0348941	0.000000
$f(X)$	1.72487	1.64226	1.724852
NFE	17425	11790	NA
Std	7.27403e-05	1.31468e-05	NA
Time (*sec*)	0.4 sec	0.3	NA

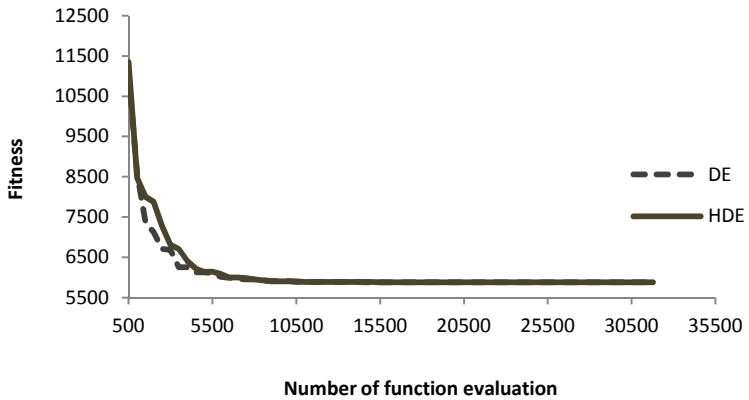


Fig. 2. Convergence graph of DE and HDE for pressure vessel design problem.

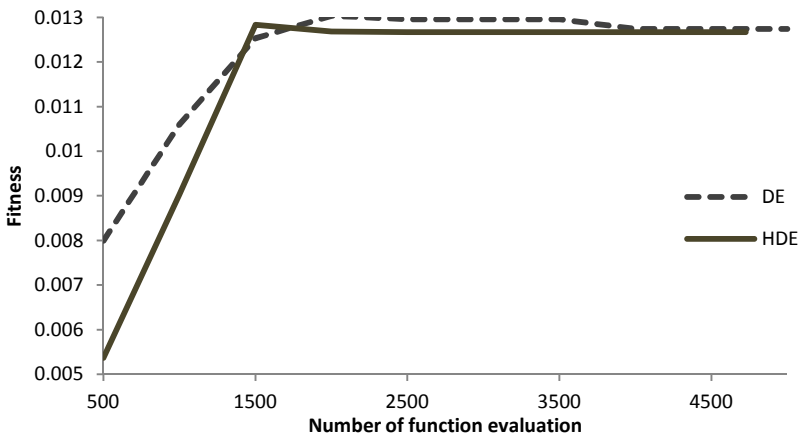


Fig. 3. Convergence graph of DE and HDE for spring design problem.

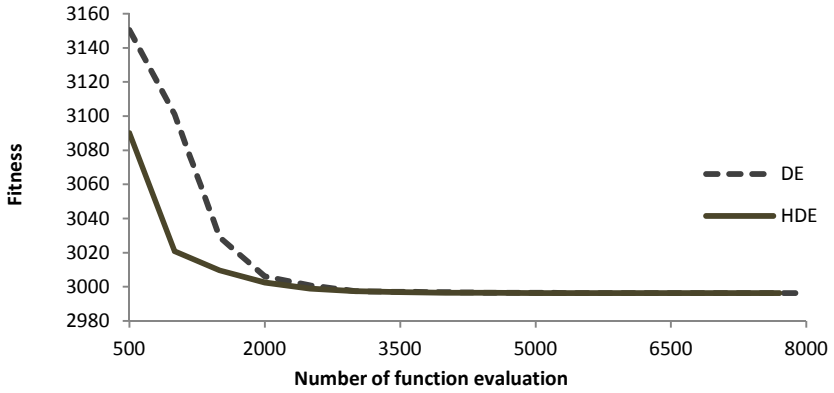


Fig. 4. Convergence graph of DE and HDE for Speed reducer design problem.

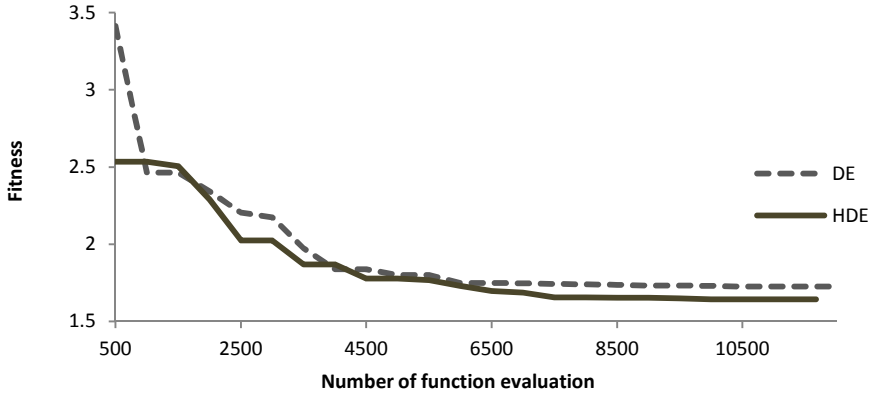


Fig. 5. Convergence graph of DE and HDE for welded beam design problem.

7. Conclusions

In the present study, we proposed a simple and easy to implement hybrid version of DE and PSO algorithm called HDE. The proposed HDE algorithm works in two stages. In the first stage DE is activated to contract the search space and when the search space has contracted enough, PSO is activated to further refine the search. The swap from DE to PSO is decided with the help of a *switchover constant* named α. We tested the proposed algorithm for scalable problems of dimensions 10 and 30 and observed that although it does not make much improvement in the fitness function value but in terms of NFE and CPU time there is an improvement of more than 20%. The superior performance of HDE is also evident from the four cases of real life problems, where it outperformed the other algorithms in terms of all the performance measures taken for the present study. We would also like to point out that we have used the basic versions of DE

and PSO. It would be interesting to note the performance of the proposed scheme while using some more advanced versions of DE and PSO algorithms.

References

1. H. Abbass, The self-adaptive pareto differential evolution algorithm, *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 831–836, 2002.
2. M. A. Abido, Optimal design of power system stabilizers using particle swarm optimization, *IEEE Transactions on Energy Conversion* 17(3), 406–413, 2002.
3. A. Abraham, S. Das, and A. Konar, Kernel based automatic clustering using modified particle swarm optimization algorithm, *2007 Genetic and Evolutionary Computation Conference*, *GECCO 2007*, ACM Press, Dirk Thierens *et al*. (Eds.), ISBN 978-1-59593-698-1, pp. 2–9, 2007.
4. A. Abraham, H. Liu, and T. G. Chang, Variable neighborhood particle swarm optimization algorithm, *Genetic and Evolutionary Computation Conference* (*GECCO-2006*), Seattle, USA, Late Breaking Papers, *CD Proceedings*, Jörn Grahl (Ed.), 2006.
5. M. M. Ali, Differential evolution with preferential crossover, *European Journal of Operation Research* 181, pp. 1137–1147, 2007.
6. J. S. Arora, *Introduction to Optimum Design*, New York, McGrow-Hill, 1989.
7. T. Back, F. Hoffmeister, and H. Schwefel, A survey of evolution strategies. In: *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, pp. 2–9, 1991.
8. C. S. Beightler, D. Phillips, and D. Wilde, *Foundations of Optimization* (Prentice-Hall, Englewood Cliffs, NJ: 1979), 2nd edn.
9. K. Bergey Paul and Cliff Ragsdale, Modified differential evolution: A greedy random strategy for genetic recombination, *Omega The International Journal of Management Science* 33, 255–265, 2005.
10. V. D. F. Bergh, Particle swarm weight initialization in multi-layer perceptron artificial neural networks, *Development and Practice of Artificial Intelligence Techniques* (Durban, South Africa, 1999), pp. 41–45.
11. T. R. Bhat, D. Venkataramani, V. Ravi, and C. V. S. Murty, An improved differential evolution method for efficient parameter estimation in biofilter modeling, *Biochemical Engineering Journal* 28, 167–176, 2006.
12. J. Brest, S. Greiner, B. Boškovic, M. Mernik, and V. Žumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Transactions on Evolutionary Computation*, Vol. 10, Issue 6, pp. 646–657, 2006.
13. L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello, Solving engineering optimization problems with the simple constrained particle optimizer, *Informatica* 32, 319–326, 2008.
14. Ji-Pyng, Chiou Chung-Fu Chang, and Ching-Tzong Su, Ant direction hybrid differential evolution for solving large capacitor placement problems, *IEEE Transactions on power systems*, 19(4), 1794–1800, 2004.
15. C. A. Coello Coello, Use of a self adaptive penalty approach for engineering optimization problems, *Computer in Industry* 41(2), 113–127, 2000.
16. S. Das and A. Konar, Two-dimensional IIR filter design with modern search heuristics: A comparative study, *International Journal of Computational Intelligence and Applications* 6(3), Imperial College Press, 2006.
17. S. Das and A. Konar, A swarm intelligence approach to the synthesis of two-dimensional IIR filters, *Engineering Applications of Artificial Intelligence* 20(8), 1086–1096, http://dx.doi.org/10.1016/j.engappai.2007.02.004, 2007.

18. S. Das, A. Abraham, and A. Konar, Adaptive clustering using improved differential evolution algorithm, *IEEE Transactions on Systems*, *Man and Cybernetics – Part A*, IEEE Press, New York, USA, 2007.
19. S. Das, A. Konar, and U. K. Chakraborty, Two improved differential evolution schemes for faster global search, *ACM-SIGEVO Proceedings of GECCO*, Washington D.C., pp. 991–998, 2005.
20. K. Deb, GeneAS: A robust optimal design technique for mechanical component design, in D. Dasgupta and Z. Michalewicz (Eds.), *Evolutionary Algorithms in Engineering Applications* (Berlin, Springer-Verlage, 1997), pp. 497–514.
21. K. Deb, An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering* 186(2/4):311–338, 2000.
22. S. Doyle, D. Corcoran, and J. Connell, Automated mirror design using an evolution strategy, *Optical Engineering* 38(2), 323–333, 1999.
23. R. C. Eberhart and X. Hu, Human tremor analysis using particle swarm optimization. In: *Proceedings of the IEEE Congress on Evolutionary Computation* (*CEC 1999*), Washington D.C., pp. 1927–1930, 1999.
24. L. Fogel, Evolutionary programming in perspective: The top-down view. In: J. M. Zurada, R. Marks, Jr., and C. Robinson (Eds.), *Computational Intelligence*: *Imitating Life* (IEEE Press, Piscataway, NJ, USA, 1994).
25. D. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning* (Addison-Wesley, 1989).
26. Z. He, C. Wei, L. Yang, X. Gao, S. Yao, R. C. Eberhart, and Y. Shi, Extracting rules from fuzzy neural network by particle swarm optimization. In: *Proceedings of IEEE Congress on Evolutionary Computation* (*CEC 1998*), Anchorage, Alaska, USA, 1998.
27. T. Hendtlass, A combined swarm differential evolution algorithm for optimization problems. In: *Proceedings of the Fourteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Lecture Notes in Computer Science, Vol. 2070. Springer-Verlag, pp. 11–18, 2001.
28. E. Himmelblau and Lasdon, *Optimization of Chemical Processes* (McGraw Hill, 1972), 2nd edn.
29. Hui-Yuan Fan and Jouni Lampinen, A trigonometric mutation operation to differential evolution, *Journal of Global Optimization* 27:105–129, 2003.
30. T, Jayabarathi Sandeep Chalasani, Zameer Ahmed Shaik, and Nishchal Deep Kodali, Hybrid differential evolution and particle swarm optimization based solutions to short term hydro thermal scheduling, *WSEAS Transactions on Power Systems* 11(2), 245–254, 2007.
31. P. Kaelo and M. M. Ali, Differential evolution algorithm with hybrid mutation*, Compt. Optim. Appl.* 37, 231–246, 2007.
32. S. Kannan, S. Slochanal, P. Subbaraj, and N. Padhy, Application of particle swarm optimization technique and its variants to generation expansion planning, *Electric Power Systems Research* 70(3), 203–210, 2004.
33. J. Kennedy and R. C. Eberhart, Particle swarm optimization, *IEEE Int. Conf. on Neural Networks* (Perth, Australia), IEEE Service Center, Piscataway, NJ, pp. 1942–1948, 1995.
34. J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, in: Pavel Ošmera (Ed.), *Proceedings of MENDEL 2000*, *6th International Mendel Conference on Soft Computing*, pp. 76–83, Brno, Czech Republic, 2000.
35. T. Masters and W. Land, A new training algorithm for the general regression neural network, *Proceedings of Computational Cybernetics and Simulation*, organized by *IEEE Systems*, *Man*, *and Cybernetics Society* 3, 1990–1994, 1997.

36. L. Messerschmidt and A. P. Engelbrecht, Learning to play games using a PSO-based competitive learning approach, *IEEE Transactions on Evolutionary Computation* 8(3), 280–288, 2004.
37. M. Omran, A. Salman, and A. P. Engelbrecht, Self-adaptive differential evolution, computational intelligence and security, PT 1, *Proceedings Lecture Notes in Artificial Intelligence* 3801: 192-199, 2005.
38. M. G. H. Omran, A. P. Engelbrecht, and A. Salman, Bare bones differential evolution, *European Journal of Operational Research,* doi:10.1016/j.ejor.2008.02.035, 2008.
39. M. Pant, M. Ali, and V. P. Singh, Differential evolution with parent centric crossover, *Second UKSIM European Symposium on Computer Modeling and Simulation*, 141–146, 2008.
40. M. Pant, M. Ali, and V. P. Singh, Parent centric differential evolution algorithm for global optimization, *OPSEARCH* 46(2), 153–168, 2009.
41. S. Paterlini and T. Krink, Differential evolution and particle swarm optimization in partitional clustering, *Computational Statistics and Data Analysis*, Vol. 50, 1220–1247, 2006.
42. S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, A novel population initialization method for accelerating evolutionary algorithms, *Computer and Applied Mathematics with Application* (53), 1605–1614, 2007.
43. S. S. Rao, *Engineering Optimization*: *Theory and Practice* (Wiley Eastern Ltd., 1996), 3rd edn.
44. T. Rogalsky, S. Kocabiyik, and R. Derksen, Differential evolution in aerodynamic optimization, *Canadian Aeronautics and Space Journal* 46(4), 183–190, 2000.
45. J. Salerno, Using the particle swarm optimization technique to train a recurrent neural model, *IEEE International Conference on Tools with Artificial Intelligence*, pp. 45–49, 1997.
46. A. Salman, A. P. Engelbrecht, and M. G. H. Omran, Empirical analysis of self adaptive differential evolution, *European Journal of Operational Research* 183, pp. 785–804, 2007.
47. E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, *J. Mech. Des.-T. ASME* 112(2):223–229, 1990.
48. H. Shen, Y. Zhu, B. Niu, and Q. H. Wu, An improved group search optimizer for mechanical design optimization problems, *Progress in Natural Science* 19, 91–97, 2009.
49. Y. Shi and R. C. Eberhart, A modified particle swarm optimiser, *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, May 4–9, 1998.
50. Y. Shi and R. C. Eberhart, Fuzzy adaptive particle swarm optimization, In: *Proceedings of the Congress on Evolutionary Computation 2001*, Seoul, Korea, IEEE Service Center, IEEE (2001), pp. 101–106, 2001.
51. R. Storn and K. Price, Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, Berkeley, CA, 1995.
52. G. Stumberger, D. Dolinar, U. Pahner, and K. Hameyer, Optimization of radial active magnetic bearings using the finite element technique and differential evolution algorithm, *IEEE Transactions on Magnetics* 36(4), 1009–1013, 2000.
53. H. Talbi and M. Batouche, Hybrid particle swarm with differential evolution for multimodal image registration. In: *Proceedings of the IEEE International Conference on Industrial Technology*, Vol. 3, pp. 1567–1573, 2004.
54. M. P. Wachowiak, R. Smolıkova, Y. Zheng, M. J. Zurada, and A. S. Elmaghraby, An approach to multimodal biomedical image registration utilizing particle swarm optimization, *IEEE Transactions on Evolutionary Computation*, 8(3), 289–301, 2004.
55. F. S. Wang and J. W. Sheu, Multi-objective parameter estimation problems of fermentation processes using high ethanol tolerance yeast, *Chemical Engineering Science* 55(18), 3685–3695, 2000.

56. W. Xu and X. Gu, A hybrid particle swarm optimization approach with prior crossover differential evolution. In: *Proceedings of GEC09* pp. 671–677, 2009.
57. Z. Yang, J. He, and X. Yao, Making a difference to differential evolution, in Z. Michalewicz and P. Siarry (Eds.), *Advances in Metaheuristics for Hard Optimization* (Springer, 2007), pp. 415–432.
58. H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, *IEEE Transactions on Power Systems* 15(4), 1232–1239, 2000.
59. D. Zaharie, Control of population diversity and adaptation in differential evolution algorithms, In D. Matousek and P. Osmera (Eds.), *Proc. of MENDEL 2003*, *9th International Conference on Soft Computing,* Brno, Czech Republic, pp. 41–46, 2003.
60. I. Zelinka and J. Lampinen, An evolutionary learning algorithms for neural networks. In: *Proceedings of Fifth International Conference on Soft Computing*, *MENDEL'99*, pp. 410–414, 1999.
61. C. Zhang, J. Ning, S. Lu, D. Ouyang, and T. Ding, A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization, *Operations Research Letters*, Vol. 37, 117–122, 2009.
62. W. J. Zhang and X. F. Xie, DEPSO: Hybrid particle swarm with differential evolution operator. In: *IEEE International Conference on Systems*, *Man*, *and Cybernetics*, Vol. 4, pp. 3816–3821, 2003.
63. Z.-F. Hao, G.-H. Gua, and H. Huang, A particle swarm optimization algorithm with differential evolution, *Proceedings of Sixth Int. Conf. on Machine Learning and Cybernetics*, pp. 1031–1035, 2007.
64. G.-N. Jose, E. Alba, and J. Apolloni, Particle swarm hybridized with differential evolution: black box optimization benchmarking for noisy functions, *Proceedings of Int. Conf. Genetic and Evolutionary Computation*, pp. 2343–2350, 2009.
65. H. Liu, Z. Cai, and Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, *Applied Soft Computing*, 10:629–640 (2009).
66. Caponio, F. Neri, and V. Tirronen, Superfit control adaption in memetic differential evolution frameworks, *Soft Computing*, 13:811–831(2009).
67. W. Xu and X. Gu, A hybrid particle swarm optimization approach with prior crossover differential evolution, *Proceedings of ACM/SIGEVO summit on Genetic and Evolutionary Computation*, pp. 671–678, 2009.
68. M. Pant, R. Thangaraj, and A. Abraham, DE-PSO: A new hybrid meta-heuristic for solving global optimization problems, *New Mathematics and Natural Computation*, accepted for publication (2009).
69. S. Khamsawang, P. Wannakarn, and S. Jiriwibhakorn Hybrid PSO-DE for solving the economic dispatch problem with generator constraints, *Proceedings of the IEEE Int. Conf. on Computer and Automation Engineering*, Vol. 5, pp. 135–139, 2010.

Appendix A. Unconstrained Benchmark Problems

1. Sphere function (SP):

$$f_{SP}(x) = \sum_{i=1}^{n} x_i^2 \quad \text{with } -5.12 \le x_i \le 5.12, \min f_{sp}(0,\ldots,0) = 0$$

2. Acley's function (ACK):

$$f_{ACK}(X) = -20 * \exp\left(-.2\sqrt{1/n\sum_{i=1}^{n} x_i^2}\right) - \exp\left(1/n\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right) + 20 + e,$$

with $-30 \le x_i \le 30$, $\min f_{ACK}(0,\ldots,0) = 0$.

3. Rastrigin's function (RG):

$$f_{RG}(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)), \text{with} -5.12 \le x_i \le 5.12, \min f_{RG}(0,\ldots,0) = 0.$$

4. Rosenbrock Problem (RB):

$$f_{RB}(x) = \sum_{i=1}^{n-1}[100(x_{i+1}^2 - x_i^2) + (1 - x_i^2)], \text{ with } -30 \le x_i \le 30, \min f_{RB}(1,\ldots,1) = 0.$$

5. Griewenk function (GW):

$$f_{GW}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \text{ with } -600 \le x_i \le 600, \min f_{GW}(0,\ldots,0) = 0.$$

6. Levy and Montalvo 2 Problem (LM2):

$$f_{LM2}(x) = \sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)(1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)(1 + \sin^2(2\pi x_n)),$$

with $-5 \le x_i \le 5$, $\min f_{LM2}(1,\ldots,1) = 0$.

7. Paviani problem (PP):

$$f_{PP}(x) = \sum_{i=1}^{n}[(\ln(x_i - 2))^2 + (\ln(10 - x_i))^2] - \left(\prod_{i=1}^{n} x_i\right)^{0.2}, \text{ with } 2 \le x_i \le 10.$$

8. Sinusoid function (SIN):

$$f_{SIN}(x) = -\left[A\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(B(x_i - z))\right], \text{ with } 0 \le x_i \le 180,$$

$\min f_{SIN}(x^*) = -(A + 1)$ where $x_i^* = (90 + z)$, $A = 2.5$, $B = 5$, $z = 30$.

9. Step function (ST):

$$f_{ST}(x) = \sum_{i=1}^{n}\left(\lfloor x_i + 0.5 \rfloor\right)^2, \text{ with } -100 \le x_i \le 100, \min f_{ST}(-0.5 \le x_i \le 0.5) = 0.$$

10. Schwefel's problem (SWF):

$$f_{SWF}(x) = 418.9829 \times n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right), -500 \le x_i \le 500, \min f_{SWF}(s,\ldots,s) = 0$$

where $s = 420.97$.

Appendix B. Real Life Problems

B.1. *Welded beam design optimization problem* (*WBD*)

The problem is to design a welded beam for minimum cost, subject to some constraints.[13] Figure 6 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B. The objective is to find the minimum fabrication cost, considering four design variables: x_1, x_2, x_3, x_4 and constraints. The optimization model is summarized below:

$$\text{Minimize:} \quad f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Subject to:

$$g_1(X) = \tau(X) - 13{,}6000 \leq 0$$

$$g_2(X) = \sigma(X) - 30{,}000 \leq 0$$

$$g_3(X) = x_1 - x_4 \leq 0$$

$$g_4(X) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$$

$$g_5(X) = 0.125 - x_1 \leq 0$$

$$g_6(X) = \delta(X) - 0.25 \leq 0$$

$$g_7(X) = 6{,}000 - Pc(X) \leq 0$$

with:

$$\tau(X) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau'(X) = \frac{6{,}000}{\sqrt{2}x_1x_2}$$

$$\tau''(X) = \frac{MR}{J}$$

$$M = 6{,}000\left(14 + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2}{2R} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{x_1x_2\sqrt{2}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(X) = \frac{504{,}000}{x_4x_3^2}$$

$$\delta(X) = \frac{65{,}856{,}000}{(30 \times 10^6) x_4 x_3^3}$$

$$Pc(X) = \frac{4.013(30 \times 10^6)\sqrt{\frac{x_3^2 x_4^6}{36}}}{196}\left(1.0 - \frac{x_3\sqrt{\frac{30 \times 10^6}{4(12 \times 10^6)}}}{28}\right)$$

with $0.1 \le x_1, x_4 \le 2.0$ and $0.1 \le x_2, x_3 \le 10.0$.

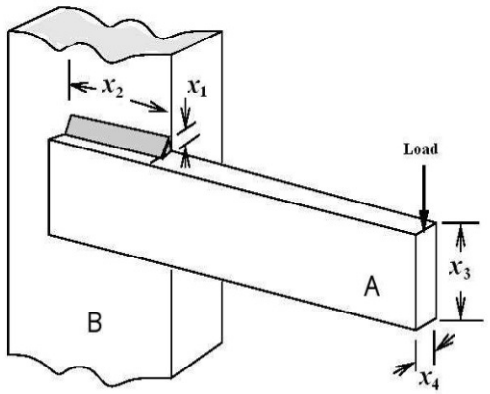


Fig. 6. Welded beam.

B.2. *Pressure vessel design optimization problem* (*PVD*)

As shown in Fig. 7, pressure vessel introduced by Sandgren[47] was designed aimed at minimizing the total cost of materials, forming and welding of the pressure vessel. There are four design variables: The shell thickness x_1, the thickness of the head x_2, the inner radius x_3 and the length of the cylindrical section x_4, x_1 and x_2 are discrete values which are integer multiples of 0.0625in, x_3 and x_4 are continuous. The pressure vessel problem is stated as follows:

Minimize: $f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$

Subject to:

$$g_1(X) = 0.0193x_3 - x_1 \le 0$$

$$g_2(X) = 0.00954x_3 - x_2 \le 0$$

$$g_3(X) = 1{,}296{,}000 - \pi x_3^2 x_4^2 - \frac{4}{3}\pi x_3^3 \le 0$$

$$g_4(X) = x_4 - 240 \le 0$$

with $0.0625 \le x_1, x_2 \le 99 \times 0.0625$ and $10.0 \le x_3, x_4 \le 200.0$.

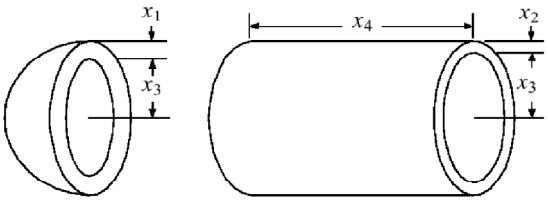


Fig. 7. Pressure vessel.

B.3. *Speed reducer design optimization problem* (*SRD*)

The design of the speed reducer[13] shown in Fig. 8, is considered with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7 (all variables continuous except x_3 that is integer). The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft. The problem is:

$$\begin{aligned}\text{Minimize: } f(X) = {} & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ & -1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\ & +0.7854(x_4x_6^2 + x_5x_7^2)\end{aligned}$$

Subject to:

$$g_1(X) = \frac{27}{x_1x_2^2x_3} - 1.0 \le 0$$

$$g_2(X) = \frac{397.5}{x_1x_2^2x_3^2} - 1.0 \le 0$$

$$g_3(X) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1.0 \le 0$$

$$g_4(X) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1.0 \le 0$$

$$g_5(X) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1.0 \le 0$$

$$g_6(X) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1.0 \le 0$$

$$g_7(X) = \frac{x_2x_3}{40.0} - 1.0 \le 0$$

$$g_8(X) = \frac{5x_2}{x_1} - 1.0 \leq 0$$

$$g_9(X) = \frac{x_1}{12x_2} - 1.0 \leq 0$$

$$g_{10}(X) = \frac{1.5x_6 + 1.9}{x_4} - 1.0 \leq 0$$

$$g_{11}(X) = \frac{1.1x_7 + 1.9}{x_5} - 1.0 \leq 0$$

with $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$ and $5.0 \leq x_7 \leq 5.5$.

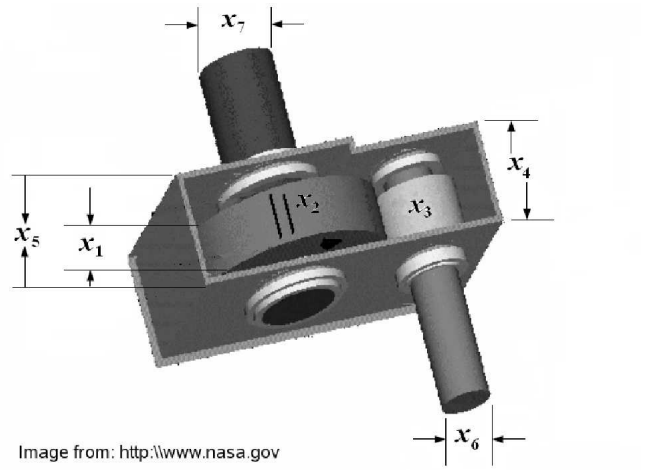


Fig. 8. Speed reducer.

B.4. *Tension/compression spring design optimization problem* (*T/CSD*)

As shown in Fig. 9, spring was designed aiming at minimizing the weight of a tension/ compression spring.[13] There are three design variables: the wire diameter x_1, the mean coil diameter x_2 and the number of active coils x_3. All design variables are continuous. The spring problem is stated as follows:

$$\text{Minimize:} \quad f(X) = x_1^2 x_2 (2.0 + x_3)$$

Subject to:

$$g_1(X) = 1.0 - \frac{x_2^3 x_3}{7{,}178 x_1^4} \leq 0$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12{,}566 x_2 x_1^3 - x_1^4} + \frac{1}{5{,}108 x_1^2} - 1.0 \leq 0$$

$$g_3(X) = 1.0 - \frac{140.45x_1}{x_2^2 x_3} \leq 0$$

$$g_4(X) = \frac{x_1 + x_2}{1.5} - 1.0 \leq 0$$

with $0.05 \leq x_1 \leq 2.0$, $0.25 \leq x_2 \leq 1.3$ and $2.0 \leq x_3 \leq 15.0$.

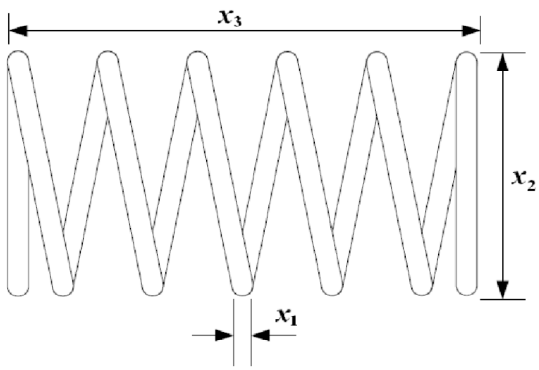


Fig. 9. Tension/compression spring.