JOURNAL OF INFORMATION, KNOWLEDGE AND RESEARCH IN ELECTRONICS AND COMMUNICATION

DIFFERENTIAL EVOLUTION ALGORITHMS FOR OPTIMIZATION PROBLEMS

[1] OM PRAKASH, [2] DR. BHAVIN S. SEDANI

[1] Assistant Professor, ECE Department, JT University, Jhunjhunu (Raj.)
[2] Head and Associate Professor, Department of Electronics and Communication , Engineering, VVP Engineering College, Rajkot - Gujarat

ABSTRACT : *The aim of this paper is to give fundamental insight into the differential evolution optimization algorithm. In this work, differential evolution algorithm has been studied to solve economic load dispatch problems of electrical power systems. DE has proven to be effective in solving many real world constrained optimization problems in different domains.*

Keywords : Differential Evolution, Constraint Optimization Problems, Unconstraint Optimization Problem And Economic Load Dispatch Problem.

INTRODUCTION

In recent years, new optimization techniques based on the principles of natural evolution, and with the ability to solve extremely complex optimization problems, have been developed. These techniques, also known as evolutionary algorithms, search for the solution of optimization problems, using a simplified model of the evolution process found in nature. Differential Evolution (DE) is one of these recently developed evolutionary computation techniques. Differential evolution improves a population of candidate solutions over several generations using the mutation, crossover and selection operators in order to reach an optimal solution. Differential evolution presents great convergence characteristics and requires few control parameters, which remain fixed throughout the optimization process and need minimum tuning.

The conventional economic load dispatch (ELD) problem of power generation involves allocation of power generation to different thermal units to minimize the operating cost subject to diverse equality and inequality constraints of the power system. This makes the economic load dispatch problem a large-scale highly nonlinear constrained optimization problem. It is therefore of great importance to solve this problem as quickly and accurately as possible. Conventional techniques offer good results, but when the search space is nonlinear and has discontinuities, these techniques become difficult to solve with a slow convergence ratio and not always seeking to the global optimal solution. New numerical methods are then needed to cope with these difficulties, specially, those with high speed search to the optimal and not being trapped in local minima [13, 15]. The economic load dispatch problem has been solved via many traditional optimization methods, including: Gradient-based techniques, Newton methods, linear programming, and quadratic programming. Most of these techniques are not capable of solving efficiently optimization problems with a non-convex, non-continuous, and highly nonlinear solution space.

HISTORY OF DIFFERENTIAL EVOLUTION

Originally developed from work done on Chebyshev's polynomial fitting problems, differential evolution finds its roots in the genetic annealing algorithms of Storn and Price. Classified as a parallel direct search method, DE achieved third place on benchmark problems at the first international contest on evolutionary optimization in 1996. Since then, the number of DE research papers increased significantly every year and differential evolution is now well-known in the evolutionary computation community as alternative to traditional Eas. The algorithm is considered to be easy to understand, simple to implement, reliable, and fast. Application areas are just as diverse as is the case for the particle swarm optimization algorithm and range from function optimization to the determination of earthquake hypocenters [2,14]. Similar to the previous section, the rest of this section first introduces the basic concepts of DE before the actual algorithm, associated algorithm parameters, and variations are discussed in more detail.

Differential evolution is a mathematical global optimization method for solving multidimensional functions. Main idea is to generate trial parameter vectors. Kenneth Price and Rainer Storn first introduced this algorithm, 1994 using vector differences for perturbing the vector population. Because differential evolution is a simple and powerful population-based stochastic search technique for solving global optimization problems over continuous spaces, and its effectiveness and efficiency have been successfully demonstrated in the last few years through a vast amount of

applications [17], it becomes one of the most satisfying methods for solving such engineering problems [2]. However, differential evolution method was originally proposed, in principle, to solve unconstrained optimization problems, hence we present a modified differential evolution with constraints handling for constrained optimization problems, i.e. the archived differential evolution [2].

Differential evolution is simple, robust and faster in optimization among the population based search algorithms that are stochastic in nature. In DE, after generating initial population randomly, a weighted difference of two randomly chosen individuals is added to a third randomly chosen individual to create a noisy random vector. Subsequently, crossover between the target and noisy random vector is carried out to give birth to offspring (trial vector). This newly generated trial vector then competes with the target vector, and the winner takes the position of target vector in next generation. Unlike genetic algorithm, differential evolution ensures that newly generated population is always better than the population in preceding generation. Differential evolution and its various improvement and modified versions of strategies have been successfully applied to many complex and non-linear application [17].

Fogel crafted a series of experiments in which finite state machines represented individual organisms in a population of problem solvers. These graphical models are used to describe the beh5vior or computer software and hardware, and so he termed his approach "Evolutionary Programming". The experimental procedure was as follows. A population of FSMs is exposed to the environment that is, the sequence of symbols that has been observed up to the current time. For each parent machine, as each input symbol is presented to the machine, the corresponding output symbol is compared with the next input symbol [3, 4]. The worth of this prediction is then measured with respect to the payoff function. After the last prediction is made, a function of the payoff for the sequence of symbols indicates the fitness of the machine or program. Offspring machines are created by randomly mutating the parents and are scored in a similar manner. Those machines that provide the greatest payoff are retained to become parents of the next generation, and the process iterates. When new symbols are to be predicted, the best available machine serves as the basis for making such a prediction and the new observation is added to the available database [6, 10].

Differential Evolution Algorithm

The differential Evolution algorithm (DE) is a population based algorithm like genetic algorithm using the similar operators; crossover, mutation and selection. The main difference in constructing better solutions is that genetic algorithms rely on crossover while differential evolution relies on mutation operators. This main operation is based on the differences of randomly sampled pairs of solutions in the population [9].

The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space [16, 17]. The DE algorithm also uses a non uniform crossover that can take child vector parameters from one parent more often than it does from other. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling the search for a better solution space. The Differential evolution algorithm is briefly given below and elaborated Fig, 1.

1. Initialize all the vector population randomly in the given upper and lower bound
2. Evaluate the fitness of each vector in the population
3. Generate a new population where each candidate individual is generated in parallel.
   i. Generate a new population vector
   ii. Perform crossover CR for target vector with its noisy vector to create a trial vector.
   iii. Evaluate the candidate.
   iv. Use the candidate in the new generation if it is at least as good the current individual.
4. Loop to 3 unless the termination criterion is met.

Differential evolution algorithms have a number of components, procedures or operators that must be specified in order to perform the required process. The most important components are presented and are explained one by one:

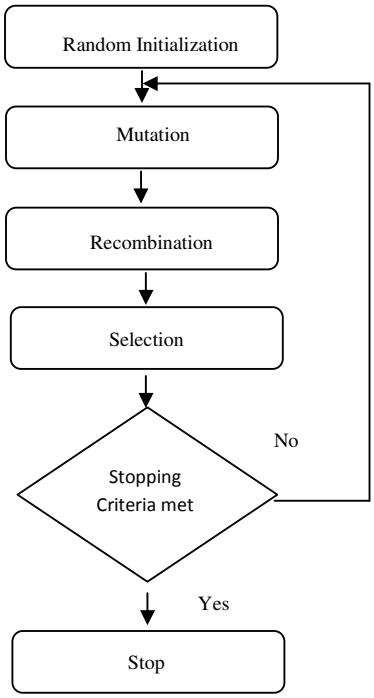

Fig 1: Flowchart for Differential Evolution

Fig 1: Flowchart for Differential Evolution

Differential evolution is population-based stochastic functions minimize (or maximize) relating to evolutionary computation, whose simple yet powerful and straightforward features make it very attractive for numerical optimization. Differential evolution uses a rather greedy and less stochastic approach to problem solving than do EAs. Differential evolution combines simple arithmetical operators with the classical operators of recombination, mutation, and selection to evolve from a randomly generated starting population to a final solution. DE differs from conventional genetic algorithms in its use of perturbing vectors, which are the difference between two randomly chosen parameter vectors, a concept borrowed from the operators of Nelder and Mead's simplex optimization technique [2, 14]. The DE algorithm was first introduced by Storn and Price in 1995 and was successfully applied in the optimization of some well-known nonlinear, non differentiable and non convex functions by Storn [14]. The different variants of DE are classified using the following notation: DE/α/β/ δ, α where indicates the method for selecting the parent chromosome that will form the base of the mutated vector, β indicates the number of difference vectors used to perturb the base chromosome, and δ indicates the recombination mechanism used to create the offspring population. The *bin* acronym indicates that the recombination is controlled by a series of independent binomial experiments. The fundamental idea behind differential evolution is a scheme whereby it generates the trial parameter vectors. In each step, the differential evolution mutates vectors by adding weighted, random vector differentials to them. If the cost of the trial vector is better than that of the target, the target vector is replaced by the trial vector in the next generation [16]. The variant implemented here was which involved the following steps and procedures.

A. Parameter tuning

The user chooses the parameters of population size Np, the boundary constraints of optimization variable x_i^{min}, x_i^{max} the mutation factor fm, the crossover rate CR, and the stopping criterion of maximum number of iterations (generations) G_{max}.

B. Initialization of an individual population

Set generation. Initialize a population of individuals (real-valued –dimensional solution vectors) with random values generated according to a uniform probability distribution in the dimensional problem space. These initial individual values are chosen at random from within User-defined bounds

$$x_{ij} = rand\ (0,1)\left(x_i^{max} - x_i^{min}\right);\ (i = 1,2,\dots,n; j = 1,2,\dots,Np\ ) \qquad (1)$$

C. Evaluation of the individual population

Evaluate the function value f(x) of each individual, at x_{ij} $(i = 1,2,\dots,n)$variable given by member of population.

D. Mutation operation (or differential operation)

Mutation is an operation that adds a vector differential to a population vector of individuals according to the following equation:

$$z_{ji}(t+1) = [x_{r1,i}(t) + f_m[x_{r2,i}(t) - x_{r3,i}(t)] \quad (2)$$
$$(i = 1,2,\dots n; j = 1,2,\dots Np)$$

where i=1, 2, 3, …, n is the individual's index of population; j=1, 2, 3, . . ., Np is the position in n-dimensional individual; t is the time(generation) $x_j(t) = [x_{j1}(t), x_{j2}(t), \dots . x_{jn}(t)]^T]$ Stands for the position of the j^{th} individual of population of Np real-valued n-dimensional vectors; $z_i(t) = [z_{j,1}(t), z_{j,2}(t), \dots, z_{j,n}(t)]^T$ Stands for the position of the J^{th} individual of mutant vector, r1, r2 and r3 are mutually different integer and also different from the running index. Randomly selected with uniform distribution from the set $\{1,2, \dots . i-1, i+1, \dots |Np\}$, and $f_m \geq 0$ is a real parameter called *mutation factor*, which controls the amplification of the difference between two individuals so as to avoid search stagnation and is usually taken from the range (0.1- 1).

E. Recombination operation

Following the mutation operation, recombination is applied to the population. Recombination is employed to generate a trial vector by replacing certain parameters of the target vector with the corresponding parameters of a randomly generated donor vector. For each vector Z_i(t+1), an index $rnbr(i)$ € {1,2,…..N}, an index is randomly chosen using uniform distribution, and a *trial vector*, U_j(t+1) = $\{U_{j,1}(t+1), U_{j,2.}$(t+1).....$U_{j,n}$(t+1)}, is generated with

$$u_{ji}(t+1) = \begin{cases} z_{j,i}(t+1) & if\ (randb(j) > CR) or\ (j = rnbr(i) \\ x_{j,i}(t) & if\ (randb(j) > CR) or\ (j \neq rnbr(i) \end{cases} \quad (3)$$

In the above equations, $randb(j)$ is the j^{th} th evaluation of a uniform random number generation with [0, 1], and is a *crossover* or *recombination rate* in the range (0 - 1). The performance of a differential evolution algorithm usually depends on three variables: the population size Np, the mutation factorF_m, and the recombination rate CR.

F. Selection operation

Selection is the procedure of producing better offspring. To decide whether or not the vector u_i(t+1), should be a member of the population comprising the next generation, it is compared with the corresponding vector x_i(t). Thus, if denotes the objective function under minimization, then

$$x_j(t+1) = \begin{cases} u_j(t+1), & if\ f(u_j(t+1)) < f(x_j(t)) \\ x_i(t), & otherwise \end{cases} \quad . \quad (4)$$

In this case, the cost of each trial vector U_j(t+1) is compared with that of its parent target vector x_j(t). If the cost, x_j(t), of the target vector is lower than that of the trial vector, the target is allowed to advance to the next generation. Otherwise, the target vector is replaced by the trial vector in the next generation [8].

G. Stopping criterion

Set the generation number for $t = t+1$ Proceed to Step 3 until a stopping criterion is met, usually G_{max}. The stopping criterion depends on the type of problem.

It is important to note that in DE mutation, crossover and selection operations continue over the course of evolution until some stopping criterion is reached. In addition, there are three control parameters in DE, i.e., the population size *Np*, the scaling factor f_m and the crossover constant *CR*.

Differential Evolution, proposed by Price and Storn in 1995, was motivated by the attempts to use Genetic Annealing to solve the Chebychev polynomial fitting problem. Genetic Annealing is a population-based, combinatorial optimization algorithm that implements a thermodynamic annealing criterion via thresholds. Although successfully applied to solve many combinatorial tasks, genetic annealing could solve the Chebychev problem satisfactorily [11]. Price modified genetic annealing by using floating-point encoding instead of bit-string one, arithmetic operations instead of logical ones, population-driven differential mutation instead of bit inversion mutation remove the annealing criterion. Storn suggested creating separate parent and children population. Differential evolution is closely related to many other multi-point derivative free search methods, such as Evolutionary strategies, genetic algorithms. Differential evolution was one of the main avenues of research in evolutionary computation in the early 1990s, including Genetic Algorithms along with Genetic Programming, and Evolution Strategies. At this time, the term "Evolutionary Computation" was invented to describe the entire field of study. Currently evolutionary programming is a wide evolutionary computing dialect with no fixed structure or (representation), in contrast with some of the other dialects. It is becoming harder to distinguish from evolutionary strategies. Some of its original variants are quite similar to the later genetic programming, except that the program. Differential Evolution technique is that of natural evolution, i.e. for a given population of individuals, the environmental pressure causes natural selection based on survival of fitness and thus causes rise in the fitness of the population [15]. Generally a quality function is to be maximized. A set of candidate solutions is randomly created initially, with in the domain of elements of functions and the candidate solution is applied to the quality function as an abstract fitness measure. Some of better candidates based on the fitness are chosen to seed the next generation by applying recombination

and/or mutation on them. Recombination is an operator applied to two or more selected candidates and results in one or more new candidates.

OPTIMIZATION PROBLEM FORMULATION

Main objective of economic power dispatch problem is to determine the optimal combination of power outputs for all generating units, which minimizes the total fuel cost of thermal power plants while satisfying load demand and operating constraints of a power system [12].

The ED problem may be expressed by minimizing the fuel cost of generator units under constraints. Depending on load variations, the output of generators has to be changed to meet the balance between loads and generation of a power system. The power system model consists of *n* generating units already connected to the system [1]. The ED problem can be expressed as: The main objective of the ED problem is to determine minimum generation cost of the generating units, according to the operating constraints of the generators and the power system limits. The simplified fuel cost function of generators represent as quadratic functions, given in equation (5).

$$f_i(x_i) = a_i + b_i x_i + c_i x_i^2 \quad (5)$$

where a_i b_i and c_i are cost coefficients of generating unit i , P_i is the real power output of generating unit $F_i(P_i)$ is the operating fuel cost of generating unit Minimizing the fuel cost function equation (6) of all generating units in the power system is the objective of ED problem which represents as

Minimize

$$F_T = \sum_{i=1}^{n} F_i(x_i) \quad (6)$$

Subject to

(i) Generating capacity constraints

$$x_i^{min} \le x_i \le x_i^{max}; \quad (i = 1,2,\dots,n)$$

(ii) Power balance constraints

$$\sum_{i=1}^{n} x_i - P_D - P_L = 0$$

Main objective of economic power dispatch problem is to determine the optimal combination of power outputs for all generating units, which minimizes the total fuel cost of thermal power plants while satisfying load demand and operating constraints of a power system [20].

The ED problem may be expressed by minimizing the fuel cost of generator units under constraints. Depending on load variations, the output of generators has to be changed to meet the balance between loads and generation of a power system. The power system model consists of *n* generating units already connected to the system. The effects of multi-valves steam turbine produce a ripple curve on quadratic fuel cost functions and represented as rectified sinusoidal function. Considering the valve point loading effects, the quadratic cost function in was modified as equation (7).

Minimize operation cost;

$$f(x_i) = \sum_{i=1}^{n} (a_i + b_i x_i + c_i x_i^2) + \left| e_i \sin\left(f_i(x_i^{min} - x_i)\right)\right| Rs/h \quad (7)$$

Subjects to;

i. Power balance constraints

$$\sum_{i=1}^{n} x_i - P_D - P_L = 0 \quad (7a)$$

and,

$$P_L = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i B_{ij} P_j + \sum_{i=1}^{n} B_{oi} x_i + B_{oo} \quad (7b)$$

ii. Generating capacity constraints

$$x_i^{min} \le x_i \le x_i^{max}; (i = 1,2,\dots,n) \quad (7c)$$

where,

a_i, b_i and c_i are the cost coefficients of the i^{th} generator

n is the number of generators

x_i is the real power output of the i^{th} generator (MW)

$f(x_i)$ is the operating cost of unit *i* (Rs/h)

P_L is the transmission losses (MW)

x_i^{max} is the maximum generation output of the i^{th} generator

x_i^{min} is the minimum generation output of the i^{th} generator

B_{ij}, B_{oi} and B_{oo} are the B-coefficients

P_D is the total demand (MW).

Differential evolution is a population based optimization algorithm that was originally developed for solving unconstrained problems. By applying an exterior penalty function we transform a constrained non-linear ED problem into an unconstrained problem. Rewrite the problem shown in (7) as

$$F_m(P_i, r_k) = \sum_{i=1}^{n} F_i(P_i) + r_k . B . h^2 \quad (8)$$

where,

r_k is penalty factor. h is the equality constrained defined as

$$h = \sum_{i=1}^{n} x_i - P_D - P_L = 0 \quad (9)$$

The goal of optimization process is to find a set of variables such that the objective function so called performance index is minimized or maximized.

Non-conventional methods like Genetic algorithm and evolutionary programming do not require gradients however it works on the fitness of the solutions in terms of goals/objective of the problem [5, 7]. The chapter describes the non-conventional methods to solve the optimization problem having single as well as multiple performance indices. The detailed stepwise implementation of Differential Evolution algorithms technique is also presented.

DE Algorithm for ELD Problem

In this section, a new approach, DE algorithm is described for solving the ELD problems.

1. Parameter tuning

For initialization, choose number of Generator units n , population size NP. Specify maximum and minimum capacity of each generator, power demand, B-coefficients matrix for calculation of transmission loss. Initialize DE parameters like Crossover Probability CR, Scaling Factorf_m. Set maximum number of Iteration.

2. Initialization of an individual population

Initialize the Population x_{ji} Since the decision variables for the ELD problems are real power generations, they are used to represent each element of a given population set. Each element of the Population matrix is initialized randomly within the effective real power operating limits. Each population set of the population matrix should satisfy equality constraint using the concept of slack generator. Each individual population set of the population matrix represents a potential solution to the given problem.

3. Find Initial feasible solution

$$x_{j,i} = x_{j,i} + w_r * \left(x_i^{max} - x_i^{min}\right) * rand\ (0,1) * h_j, \qquad (i = 1,2 \dots, n\,; j = 1,2 \dots, Np) \quad (10)$$

where

$$\boldsymbol{h_j} = \sum_{i=0}^{N} x_{j,i} - P_D - P_{Lj} \qquad (10a)$$

4. Evalution of function

$$f_j\left(x_{ji}, r_k\right) = \sum_{i=1}^{n} f_{ji}(x_{ji}) + r_k\, h_j^2 \qquad (11)$$

After fitness operation, new offspring population x set is generated. In ELD problems these represent new modified generation values of generators (x). Equality constraint is satisfied using concept of slack generator. Fitness value of each newly generated population set (offspring matrix P') is recomputed, i.e., fuel cost of each power generation set. Perform selection operation between parent population (x) and newly generated (x') offspring based on their fitness values. This loop can be terminated after a predefined number of iteration

5. Mutation

Mutation is an operation that adds a vector differential to a population vector of individuals according to the following equation:

$$z_{ji}(t+1) = [x_{r1,i}\ (t) + f_m\big[x_{r2,i}(t) - x_{r3,i}(t)] \qquad (12)$$

$$(i = 1,2, \dots n; j = 1,2, \dots Np)$$

where i=1, 2, 3, …, n is the individual's index of population; j=1, 2, 3, . . ., Np is the position in n-dimensional individual.

6. Recombination operation

Following the mutation operation, recombination is applied to the population. Recombination is employed to generate a trial vector by replacing certain parameters of the target vector with the corresponding parameters of a randomly generated donor vector. For each vector Z_i(t+1), an index $rnbr(i)$ € {1,2,…..N}, an index is randomly chosen using uniform distribution, and a *trial vector*, U_j(t+1) = $\{U_{j,1}(t+1), U_{j,2.}$(t+1).....$U_{j,n}$(t+1)}, is generated with

$$u_{ji}(t+1) \begin{cases} z_{j,i}(t+1) & if\ (randb(j) > CR) or\ (j = rnbr(i) \\ x_{j,i}(t) & if\ (randb(j) > CR) or\ (j \neq rnbr(i) \end{cases} \quad (13)$$

In the above equations, $randb(j)$ is the j^{th} th evaluation of a uniform random number generation with [0, 1], and is a *crossover* or *recombination rate* in the range (0 - 1). The performance of a differential evolution algorithm usually depends on three variables: the population size Np, the mutation factor F_m, and the recombination rate CR

.7. Selection operation

Selection is the procedure of producing better offspring. To decide whether or not the vector u_i(t+1), should be a member of the population comprising the next generation, it is compared with the corresponding vector x_i(t). Thus, if denotes the objective function under minimization, then

$$x_j(t+1) = \begin{cases} u_j(t+1), & if\ f(u_j(t+1)) < f(x_j(t)) \\ x_i(t), & otherwise \end{cases} \quad . \qquad (14)$$

In this case, the cost of each trial vector U_j(t+1) is compared with that of its parent target vector x_j(t). If the cost, x_j(t), of the target vector is lower than that of the trial vector, the target is allowed to advance to the next generation. Otherwise, the target vector is replaced by the trial vector in the next generation [9].

8. Stopping criterion

Set the generation number for $t = t + 1$ Proceed to Step 3 until a stopping criterion is met, usually G_{max}. The stopping criterion depends on the type of problem.

Results and discussion

The DE algorithm is developed in Fortran 77, objects oriented language and has been successfully tested on benchmark test problems, which involve constraints on state variables and control variables. The obtained results are widely

affected by the search area boundaries, number of iteration and tuning of parameter. The different benchmark test examples undertaken for study are presented in the ensuing section.

UNCONSTRAINTS OPTIMIZATION PROBLEM :

A benchmark test problem is defined below, and is solved by differential evolution optimization algorithm.

Test Problem1. (A Quadratic Function). This Problem is defined by

Minimize $f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$

The global minimize is x^*= (1, 3) with $f(x_1^*, x_2^*) = 0$.where x^* is optimal point

In this problem the different parameters to implement differential evolution are considered such as penalty coefficient, r_k is set to 100.0, mutation factor, f_m is taken as 0.45 and crossover factor, CR is adjusted 0.995, and the population size Np is set to 60.

Table 1: Result of DE for unconstraint optimization problem

S.	Iteratio	x_1^*	x_2^*	$f(x_1^*, x_2^*)$
1	20	0.943380	2.586125	1.059959
2	30	0.95227	2.60248	0.953290
3	40	0.96218	2.61228	0.87529
4	50	0.96218	2.71228	0.50811
5	60	0.96412	2.79635	0.27225
6	70	0.97142	2.83145	0.184667
7	80	0.98852	2.90976	0.049663
8	90	0.99527	2.96456	0.007732
9	100	0.99527	2.9992	0.000143
10	110	0.99889	2.9992	0.000016

The best solution is obtained by implementation differential evolution algorithm explained. The obtained control variable are x_1^* = 0.99889, x_2^* =2.9992 and $f(x_1^*, x_2^*)$ = 0.000016. The effect of number of iterations while obtaining the result has realized through Fig.1. It can be concluded that small iterations terminates prematurely.

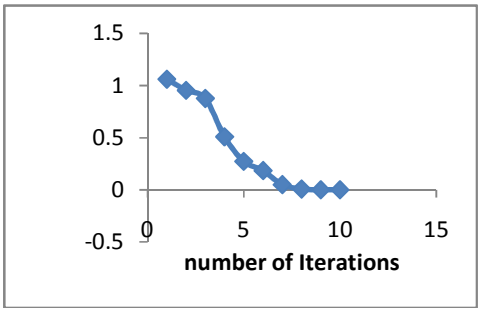

Fig.1: Variation of function with respect to iteration for unconstraint problem 1.

Test Problem 2; (Beale's Function). This Problem is defined by

$f(x_1, x_2) = [1.5\text{-}x_1(1 - x_2)]^2 + [2.25\text{-}x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$

The global minimize is x^*= (3, 0.5) with $f(x_1^*, x_2^*)$=0.0

In this problem the different parameters to implement differential evolution are considered such as penalty coefficient, r_k is taken as 100.0, mutation factor, f_m is set to 0.45 and Crossover factor, CR is adjusted 0.995, and the population size Np is set to 60.

Table 2: Result of DE for unconstraint optimization problem 2.

S.	Iteration	x_1^*	x_2^*	$f(x_1^*, x_2^*)$
1	20	2.97149	0.443764	0.04948
2	30	2.97149	0.047639	0.00598
3	40	2.97989	0.48456	0.00245
4	50	2.98546	0.49152	0.000562
5	60	2.99128	0.49152	0.000910
6	70	2.99897	0.49152	0.000037
7	80	2.99895	0.49854	0.00003

The best solution is obtained. The obtained control variable are x_1^* = 2.99895, x_2^* =0.49854 and $f(x_1^*, x_2^*)$ = 0.00003. The effect of number of iterations while obtaining the result has been realized through Fig. 2. It can be concluded that small iterations terminates prematurely

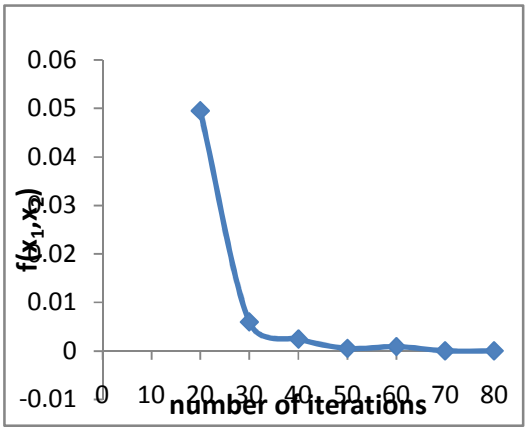

Fig. 2: Variation of function with respect to iteration for unconstraint problem 2.

ECONOMIC LOAD DISPATCH PROBLEM

The classical economic dispatch problem is an optimization problem that determines the power output of each online generator that will result in a least cost system operating state [8]. The ELD problem can then be written in the following form can be described as an optimization (minimization) process with the objective:

$$Minimized \quad f = \sum_{i=1}^{n} F_i\ (x_i)$$

Subject to:

(i) Power balance constraints:

$$P_D = \sum_{i=1}^{n} x_i$$

(ii) Generating capacity constraints:

Generating units have lower (x_i^{min}) and upper (x_i^{max}) production limits,

$$x_i^{min} \le x_i \le x_i^{max} \quad (i = 1, 2, \ldots, n) \quad (4.4c)$$

The fuel-cost function without valve-point loadings of the generating units is given by

$$F_i(x_i) = (a_i x_i^2 + b_i x_i + c_i) \quad (4.4d)$$

A cost function is obtained based on the ripple curve for more accurate modeling which contains higher order nonlinearity and discontinuity due to the valve point effect and should be refined by a sine function [19, 20]. Therefore, the fuel cost function can be expressed as

$$F_i(x_i) = (a_i x_i^2 + b_i x_i + c_i) + \left|e_i \sin\left(f_i(x_i^{min} - x_i)\right)\right| \quad (4.4e)$$

\ POWER SYSTEM ECONOMIC LOAD DISPATCH STUDY

In this section, the proposed approach was tested with two standard load dispatch problems (3-generator). The iteration was varied and the penalty multiplier was taken as 1000 for all selected methods.

3- GENERATORS ELECTRICAL POWER SYSTEM

A system of 3-Generators with the effects of valve-point loading was studied in this test. In this case, the load demand expected to be determined as P_D=850 MW [6]. Based on data, Other different parameters to implement differential evolution are considered such as penalty coefficient, r_k is set to 10000.0, mutation factor f_m is adjust to 0.45 and crossover factor, CR is taken as 0.995, population size, Np is set to 100. The three generators test is represented in Tables 3, which show that the differential evolution succeeded in finding the satisfactory solution as compared to other optimization method.

Table 3: 3-Generator electrical power system's characteristics

Generator number	Generator limits		Fuel cost coefficients				
	x_i^{min} (MW)	x_i^{max} (MW)	a_i (Rs/MW^{2}h)	b_i (Rs/MWh)	c_i (Rs/h)	e_i (Rs/h)	f_i (MW^{-1})
1	100	600	0.001562	7.92	561	300	0.031
2	50	200	0.004820	7.97	78	150	0.063
3	100	400	0.001940	7.85	310	200	0.042

Economic load dispatch problem has been solved using differential evolution for 3-Generators sample system when demand is 850 MW. The variation of operating cost with number of iteration is given in Fig. 4. The generation schedule is given in Table 5.

Table 4: Cost at different iterations for 3-Generator power system

Generator number	Generator limits		Fuel cost coefficients				
	x_i^{min} (MW)	x_i^{max} (MW)	a_i (Rs/MW^{2}h)	b_i (Rs/MWh)	c_i (Rs/h)	e_i (Rs/h)	f_i (MW^{-1})
1	100	600	0.001562	7.92	561	300	0.031
2	50	200	0.004820	7.97	78	150	0.06

							3
3	100	400	0.001940	7.85	310	200	0.042

Economic load dispatch problem has been solved using differential evolution for 3-Generators sample system when demand is 850 MW. The variation of operating cost with number of iteration is given in Fig. 3. The generation schedule is given in Table 5.

Table 5: Cost at different iterations for 3-Generator power system

S.No.	**Iterations**	**Operating cost (Rs/h)**	x_1 (MW)	x_2 (MW)	x_3 (MW)
1	20	8489.937	496.2212	194.6662	159.1041
2	30	8397.72	403.0955	120.2685	326.0829
3	40	8438.748	502.6844	089.8426	257.6016
4	50	8339.112	495.426	99.3815	255.8299
5	60	8295.868	502.7097	98.170	248.799
6	70	8258.31	398.410	200.000	250.000
7	80	8251.466	498.541	101.5429	249.866
8	90	8249.233	498.541	101.5429	249.866
9	100	8242.482	501.784	102.200	245.83

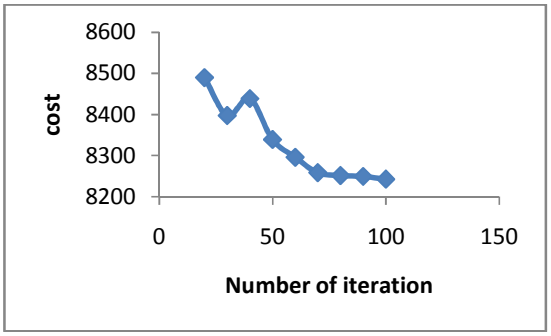

Fig. 3: Variation of function with respect to number of iteration

CONCLUSION

Differential evolution appears to be the most efficient in terms of faster convergence rate and quality of solution, which makes it to be much efficient in finding the global optimum. More the population size, better the results. Higher iteration give better results subject to the tuning of the parameter like fm, CR. And Np.

Reference

1. Amjady, N. and Nasiri-Rad, H. [2009], "Economic dispatch using an efficient real coded genetic algorithm", IET Generation, Transmission and Distribution, vol. 3, no 6, pp. 266 –278.
2. Das, S. Abraham, A. and Konar, A. [2007], "Adaptive Clustering Using Improved Differential Evolution Algorithm" IEEE Transactions on Systems, New York, US
3. Fogel, L.J. and Burgin, G.H. [1969], "Competitive goal-seeking through evolutionary programming", Final Report, Contract AF 19(628)-5927, Air Force Cambridge Research Laboratories.
4. Fogel, Gary B. Greenwood, Garrison W. and Kumar, Chellapilla. [2000], "Evolutionary computation with extinction: Experiments and analysis", Proceedings of the Congress on Evolutionary Computation, vol.2, pp. 1415-1420.
5. Gopal, M. [1984], "Modern Control Theory", Wiely Eastern Limited.
6. Gnanadass, R. Venkatesh, P. and Padhy, N. P. [2005], "Evolutionary programming based optimal power flow for units with non-smooth fuel cost functions", *Electric Power Component and System*, vol. 33, pp. 349–361.
7. Gaing, Z.L. [2003], "Particle swarm optimization to solving the economic dispatch considering the generator constraints", *IEEE Transcation on Power System*, vol. 18, no. 3, pp 1187–1195.
8. Kuo , C. C. [2008], "A novel coding scheme for practical economic dispatch by modified particle swarm approach", *IEEE Transcation on Power System.,* vol. 23, no. 4, pp. 1825–1835.
9. Lampinen, J. [2008], "Multi-Constrained Nonlinear Optimization by the Differential Evolution Algorithm",Czech Republic,pp. 76–83.ISBN 80-214-1609-2.
10. Mendes Rui, James Kennedy and Jose'Neves,[2004] The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation, vol.8no. 3, pp.204-210, June 2004
11. Mishra, S.K.: [2006] "Global Optimization by Differential Evolution and Practical swarm optimization Method Evolution on Some Benchmark Function" SSRN.
12. Nomana, N. and Iba, H. [2008], "Differential evolution for economic load dispatch problems,"*Electric. Power System Research*, vol. 78, no. 3, pp. 1322–1331.
13. Price, K.V., Storn, R.M., Lampinen, J.A. [2005] , "Differential Evolution: A Practical Approach to Global Optimization", Springer, Berlin, Heidelberg.
14. Pérez-Guerrero, R. E. and Cedeño-Maldonado, J. R. [2005] , "*Differential Evolution Based*

economic environmental power dispatch", North American Power Symposium 23-25, pp. 191- *197.*

15. Swain, Morris. Kumar Anjan. and Alan, S. [2000] , "A novel hybrid Evolutionary Programming method for function optimization", Proceedings of the 2000 Congress on Evolutionary Computation, vol.1, pp.699-705, NY.

16. Storn, R. [1997], "Differential evolution—a simple and efficient heuristic for Global over continuous spaces", *Jaunal of Global Optimization*, vol. 11, no. 4, pp. 341–359.

17. Salman, A. Engelbrecht, A.P. and Omran, M.G.H. [2007]. "Empirical analysis of self-adaptive differential evolution", European Journal of Operational Research, vol.183, no. 2, pp. 785-804.

18. Victoire, T.A.A. and Jeyakumar, A.E. [2004], " Hybrid PSO-SQP for economic dispatch with valve-point effect, Electric Power System Research", vol.71 ,no 1, pp 51–59.

19. Victoire ,T.A.A. and Jeyakumar, A.E. [2004] ,"Discussion of Particle swarm optimization to solving the economic dispatch considering the generator constraints," IEEE Transcation on Power Systems, vol.19, no. 4, pp.2121-2123.

20. Wang, S.K. Chiou, J.P. and Liu, C.-W. [2007], "Non-smooth/non-convex economic dispatch by a novel hybrid differential evolution algorithm", *IET* Generation Transmission and Distributio, vol. 1, no. 5, pp. 793–803.