# Economic Emission Load Dispatch using Interval Differential Evolution Algorithm

Archana Gupta
*Department of Electrical, B.I.T., Durg - 491001, India, archana_gupta74@rediffmail.com*
Shashwati Ray
*Department of Electrical, B.I.T., Durg - 491001, India, shashwatiray@yahoo.com*

**Abstract.** Differential Evolution (DE) algorithm is a heuristic approach for minimizing nonlinear and non-differentiable continuous space functions (storn, 1997). However, the implementation of DE requires some estimate of the global minimum to be provided, which is difficult for any arbitrary function. Also, the termination with maximum number of iterations does not assure the global minima. Moreover, as the region of global minimum approaches, the convergence of the algorithm is very slow. In order to overcome these drawbacks, we propose here a Modified Differential Evolution algorithm (MDEI) which uses interval analysis (Moore, 1966) in three phases. In the first phase of MDEI, global minimum is roughly estimated with the application of interval arithmetic, which provides the upper and lower bounds of the objective function. In the second phase of MDEI, an interval branch and bound algorithm is used to initialize a potential population. In the third phase, MDEI constructs a mechanism using interval arithmetic (Vrahatis, 1997), which updates the bounds in each generation, and is able to define an efficient termination criterion. When the criterion is fulfilled, the algorithm converges to the global minimum with certainty with a lesser computational effort.

The proposed algorithm is applied to economic emission load dispatch (EELD) problem. The purpose of EELD is to obtain the optimal amount of generated power for the fossil based generating unit in the system by simultaneously minimizing the fuel and emission costs (Dhillon, 1993). We choose IEEE 30 bus, six generator system as a test system and obtain lesser fuel and emission costs with lesser computational effort as compared to conventional Differential Evolution algorithm.

**Keywords:** Interval arithmetic, Branch and bound algorithm, Differential Evolution, Shrinking box, Fuel cost, Emissions.

## 1. Introduction

Evolutionary algorithms (EAs) are a class of population based stochastic optimization algorithms that incorporate mechanisms from evolution for optimization processes. Some well-established and commonly used EAs are Genetic Algorithms (GA) (Goldberg, 1989), Evolution Strategies (ESs) and Differential Evolution (DE) (storn, 1997).

Differential Evolution is a heuristic approach to solve global optimization problems. It is a population based method and an improved version of GA using similar operators: mutation, crossover and

*4th International Workshop on Reliable Engineering Computing (REC 2010)*
*Edited by* Michael Beer, Rafi L. Muhanna and Robert L. Mullen

ISBN: 978-981-08-5118-7. *Published by* Research Publishing Services.
doi:10.3850/978-981-08-5118-7_034

selection. The main difference in constructing better solutions is that GAs rely on crossover while DE relies on mutation operation. The mutation operation is used as a search mechanism, which is based on the differences of randomly sampled pairs of solutions in the population. The algorithm uses selection operation to direct the search towards the prospective regions in the search space. Thus DE is a simple and efficient direct search algorithm for global optimization over continuous spaces. It is extremely robust in locating the global minimum (storn, 1997).

One of the important aspects of any algorithm is the termination criterion. For theoretical aspects of evolutionary algorithms stopping criteria are usually not important. However, for practical applications the choice of stopping criteria can significantly influence the duration of an optimization run as well as the global minimum. DE terminates either with user defined number of generations or some error estimate. These criteria are perfectly suitable for comparing the performance of different algorithms, but for solving real world problems there are some drawbacks.

With predefined number of generations, optimization run might be terminated before the population has converged, or computational resources might be wasted because the optimization run is terminated late. Real world problems mostly contain computationally expensive objective functions that may result in optimization runs that take several days. Thus wasting of computational resources has to be prevented. Also this criterion is highly dependent on the objective function. Generally no correlation can be seen between an optimization problem and the required number of function evaluations, maximum number of function evaluations has to be usually determined by trial-and-error methods. The second criterion requires optimum value which is not possible for an arbitrary function. The other drawback of DE is the slow convergence near the region of global minimum.

As the DE algorithm includes randomness in the optimization process, the number of function evaluations that is needed for convergence is subject to fluctuations. Also it would be better to use stopping criteria that consider knowledge from the state of the optimization run. Several stopping criteria are reviewed in (Zielinski, 1990), that are sensitive to the state of the optimization run by observing the improvement, movement or distribution of the population members.

In this paper a Modified Differential Evolution algorithm using intervals (MDEI) is presented which uses interval analysis in three phases and is applied to Economic Emission load dispatch problem (EELD). In the first phase of MDEI the upper and lower bounds of the objective function are formed using interval arithmetic (Moore, 1966). In second phase the interval branch and bound algorithm is applied to initialize the population (Hansen, 1992). In this way the random initialization of population is replaced by a directed population. In the third phase, a mechanism is constructed using interval arithmetic that updates the bounds in each generation, and is able to define an efficient termination criterion. When the criterion is fulfilled, the algorithm converges to the global minimum with certainty. Also the next generation individuals are selected from the subset of current population, which performs with in the current upper and lower bounds of the objective function. In this way the extra computational effort is avoided.

The purpose of EELD problem is to obtain the optimal amount of generated power for the fossil based generating unit in the system by minimizing the fuel cost and emission level simultaneously, subject to various equality and inequality constraints of the power system. Different techniques have been reported in the literature pertaining to economic emission load dispatch problem. Abido had pioneered this research by applying three algorithms viz., Nondominated Sorting Genetic Algorithm (NSGA), Niched Pareto Genetic Algorithm(NPGA) and Strength Pareto Evolutionary Algorithm

(SPEA) (Abido, 1970) to EELD. Also an ant colony optimization (ACO) meta-heuristic method (Slimani, 2007) and Genetic algorithm (KORIDAK, 2008) have been applied to EELD problem.

In this paper the multi objective EELD problem is converted into a single objective problem by linear combination of different objectives as a weighted sum and then MDEI is applied. The standard IEEE 30-bus 6 generator is considered as a test system. The system is considered as lossless. The results obtained are compared with the conventional DE and henceforth the effectiveness of MDEI to solve the EELD problem is demonstrated.

This paper is organized as follows: In section 2 we give the basics of Differential Evolution algorithm and interval arithmetic. In section 3 we present our modified Differential Evolution algorithm using intervals (MDEI). In section 4 we formulate the EELD problem and define its objective. In section 5 we apply MDEI to EELD problem and compare the results with traditional DE. We conclude our study in section 6.

## 2. Basics of Differential Evolution Algorithm and Interval arithmetic

### 2.1. Differential Evolution Method (storn, 1997)

DE is a direct search method using operators: mutation, crossover and selection. The algorithm randomly chooses a population vector of fixed size. During each iteration of algorithm a new population of same size is generated. It uses mutation operation as a search mechanism. This operation generates new parameter vector by adding a weighted difference vector between two population members to a third member. In order to increase the diversity of the parameter vectors, the crossover operation produces a trial vector which is a combination of a mutant vector and a parent vector. Then the selection operation directs the search toward the prospective regions in the search space. In addition, the best parameter vector is evaluated for every generation in order to keep track of the progress that is made during the minimization process. The above iterative process of mutation, crossover and selection on the population will continue until a user-specified stopping criterion, normally, the maximum number of generations or the maximum number of function evaluations allowed is met. The process is assumed to have converged if the difference between the best function values in the new and old population, and the distance between the new best point and the old best point are less than the specified respective tolerances. The other type of stopping criterion could be if the global minimum of the problem is know a-priori. Then DE will be terminated if the difference between the best function value in the new population and the known global minimum is less than the user defined tolerance level (storn, 1997).

#### 2.1.1. *Differential Evolution optimization process*

DE optimization process operates on population $\mathbf{P}^{(G)}$ of constant size $NP$ consisting of $n$ dimensional real-valued vectors $\mathbf{x}_i^{(G)}$ given by.

$$\mathbf{P}^{(G)} = \left[\mathbf{x}_1^{(G)}, \mathbf{x}_2^{(G)}, ..., \mathbf{x}_{NP}^{(G)}\right], G = 1, ..., G_{\max} \tag{1}$$

$G$ is the generation or iteration of the algorithm to which the population belongs, $G_{\max}$ is the maximum number of generations defined by the user and

$$\mathbf{x}_i^{(G)} = \left(x_{i1}^{(G)}, x_{i2}^{(G)}, ..., x_{in}^{(G)}\right)^T, i = 1, 2, ..., NP \tag{2}$$

Also the parameters $x_{ij}$ are subjected to lower and upper boundary constraints $x_{ij}^{(L)}$and $x_{ij}^{(U)}$, respectively for $j = 1, 2, ..., n$. The population is initialized randomly by assigning random values to each decision parameter of each individual of population.

$$\mathbf{x}_i^{(G)} = x_{ij}^{(L)} + rand\,[0,1] * \left(x_{ij}^{(U)} - x_{ij}^{(L)}\right) \tag{3}$$

where $rand\,[0,1]$ denotes a uniformly distributed random value within $[0,1]$ that is new for each $j$. The population is improved by applying mutation, crossover and selection operators. The mutation operator is in charge of introducing new parameters in to the population. Two vectors $\mathbf{x}_{r2}^{(G)}$and $\mathbf{x}_{r1}^{(G)}$ are randomly selected from the population and the vector difference between them is established. This difference is multiplied by a scaling factor $\mu$, where $\mu$ is real constant $\in [0,1]$ (specified at the start and remains unchanged throughout the algorithm) and added to a third randomly chosen vector $\mathbf{x}_{r3}$ from the population. This is known as the differential variation and a mutant vector is generated as

$$\mathbf{v}_i^{(G+1)} = \mathbf{x}_{r3}^{(G)} + \mu\left(\mathbf{x}_{r2}^{(G)} - \mathbf{x}_{r1}^{(G)}\right) \tag{4}$$

Following the mutation operation, the crossover operator creates the trial vectors, which are used in the selection process. A trial vector is a combination of a mutant vector and a parent (target) vector which is formed based on probability distributions. The purpose of crossover is to inject diversity into the original population in order to avoid being trapped in the local minimum, i.e., searching the entire solution space, including unvisited areas in order to generate solutions that differ from the previous ones. The crossover probability is determined by the user defined crossover constant $CR \in [0,1]$. For each parameter, a random value based on binomial distribution is generated in $[0,1]$ and is compared against $CR$. If the value of the random number is less than or equal to the value of the $CR$ the parameter would be the mutant vector, otherwise the parameter would be the parent vector.

$$u_{ij}^{(G+1)} = \begin{cases} v_{ij}^{(G+1)} if\,(rand\,[0,1] \leq CR)\,or\,(j = rnbr\,(i)) \\ x_{ij}^{(G)} otherwise \end{cases} \tag{5}$$

where $rnbr\,(i)$ is a randomly chosen index $\in \{1, ..., n\}$ which ensures that each individual trial vector, $\mathbf{u}_i^{(G+1)}$, differs from its counterpart in the previous generation $\mathbf{x}_i^{(G)}$ by at least one parameter.

The selection operator chooses the vectors that are going to compose the population in the next generation. These vectors are selected from the current population and the trial population. Each individual of the trial population is compared with its counterpart in the current population. Assuming that the objective function is to be minimized, the vector with the lower objective function value wins a place in the next generation's population. As a result, all the individuals of the next generation are as good or better than their counterparts in the current generation.

$$\mathbf{x}_i^{G+1} = \begin{cases} \mathbf{u}_i^{(G+1)} iff\left(\mathbf{u}_i^{(G+1)}\right) < f\left(\mathbf{x}_i^G\right) \\ \mathbf{x}_i^G otherwise \end{cases} \tag{6}$$

In boundary constrained problem it is essential to ensure that parameter values lie inside their allowed ranges. A simple way to replace parameter values that violate boundary constraints with random values generated within the feasible range is as follows (Lampinen, 1999)

$$u_{ij}^{(G+1)} = \begin{cases} x_{ij}^{(U)} + rand\,[0,1] * \left(x_{ij}^{(G)} - x_{ij}^{(U)}\right) if u_{ij}^{(G+1)} > x_{ij}^{(U)} \\ x_{ij}^{(L)} + rand\,[0,1] * \left(x_{ij}^{(G)} - x_{ij}^{(L)}\right) if u_{ij}^{(G+1)} < x_{ij}^{(L)} \\ u_{ij}^{(G+1)} otherwise \end{cases} \tag{7}$$

### 2.2. DE Algorithm

**Algorithm DE** $[f^*, \mathbf{x}^*]$:= D_E$\left(\mathbf{x}^{(L)}, \mathbf{x}^{(U)}, n, f, \mu, CR, NP, G_{\max}\right)$

Inputs : lower and upper boundaries $\mathbf{x}^{(L)}$and $\mathbf{x}^{(U)}$respectively, dimension $n$, function $f$, scaling factor $\mu$, the crossover constant $CR$, the population size $NP$, maximum number of generations $G_{\max}$

Outputs : global minimum $f^*$ and real vector $\mathbf{x}^*$ of $n$ dimension at which $f^*$ occurs

BEGIN algorithm

1. {Initialization}

   Create an initial population

2. Evaluate each individual in the population

3. Find out the vector with the lowest cost

4. **while** the termination criterion not reached **do**

   - Mutation
   - Crossover
   - Selection
   - Evaluate the new individuals.

5. {Return}

   Output $f^*$and $\mathbf{x}^*$

END algorithm

### 2.3. Interval arithmetic (Moore, 1966)

Interval arithmetic is an arithmetic defined on sets of intervals, rather than on sets of real numbers. It has been invented by R. E. Moore. The power of interval arithmetic lies in its implementation on computers. In particular, outwardly rounded interval arithmetic allows rigorous enclosures for the ranges of operations and functions. This makes a qualitative difference in scientific computations, since the results are now intervals in which the exact result must lie. It has been used recently for solving ordinary differential equations, linear systems, global optimization, etc.

Let $\mathbf{x} = \{[a,b] \,|a \leq b, a, b \in \mathbb{R}\}$ be a real interval, where $a$ is the infimum and $b$ is the supremum of $\mathbf{x}$. The width of interval is defined as $w(\mathbf{x}) = b - a$. The midpoint of the interval is defined as $m(\mathbf{x}) = (a+b)/2$. For an $n$ dimensional interval vector or box $\mathbf{x}^n = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, the width of $\mathbf{x}^n$ is given by $w(\mathbf{x}^n) = [w(\mathbf{x}_1), w(\mathbf{x}_2), .., .w(\mathbf{x}_n)]$. The midpoint of $\mathbf{x}^n$ is $m(\mathbf{x}^n) = [m(\mathbf{x}_1), m(\mathbf{x}_2), ..., m(\mathbf{x}_n)]$. Let $\mathbf{x} = [a,b]$ and $\mathbf{y} = [c,d]$ be two intervals. Let +, -, * and / denote the operation of addition, subtraction, multiplication and division, respectively. If $\otimes$ denotes any of these operations for the arithmetic of real numbers $x$ and $y$, then the corresponding operation for arithmetic of interval numbers $\mathbf{x}$ and $\mathbf{y}$ is

$$\mathbf{x} \otimes \mathbf{y} = \{\mathbf{x} \otimes \mathbf{y} : x \in \mathbf{x}, y \in \mathbf{y}\} \tag{8}$$

The above definition is equivalent to the following rules:

$$\mathbf{x} + \mathbf{y} = [a + c, b + d] \tag{9}$$

$$\mathbf{x} - \mathbf{y} = [a - d, b - c] \tag{10}$$

$$\mathbf{x}.\mathbf{y} = [min\{ac, ad, bc, bd\}, max\{ac, ad, bc, bd\}] \tag{11}$$

$$\mathbf{x}/\mathbf{y} = [a,b]\,[1/d, 1/c]\, if 0 \notin \mathbf{y} \tag{12}$$

An interval function $F(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ of intervals $\mathbf{x}_1, ..., \mathbf{x}_n$ is an interval valued function of one or more variables. $F(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ is said to be an interval extension of a real function $f(x_{1,}x_2, ..., x_n)$ if $f(x_{1,}x_2, ..., x_n) \in F(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, whenever $x_i \in \mathbf{x}_i$ for all $i = 1, 2, ..., n$. $F$ is said to be inclusion monotonic if

$$x_i \subset y_i \Longrightarrow F(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n) \subset F(\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n)$$

Also $F(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ contains the range of $f(x_{1,}x_2, ..., x_n)$.

Interval functions $F(\mathbf{x})$ can be constructed in any programming language in which interval arithmetic is simulated or implemented via natural interval extensions. However, computing an interval bound carries a cost of 2 to 4 times as much effort as evaluating $f(\mathbf{x})$ (Hansen, 1992; Moore, 1966).

## 3. Proposed Modified Differential Evolution Algorithm

As discussed in section 2, DE algorithm terminates either with user defined number of generations or the implementation of the algorithm requires some estimate of the global minimum to be provided, which is not possible for any arbitrary function. Also DE converges slowly as the region of global minimum approaches.

Looking in to the above mentioned drawbacks, we propose here a Modified Differential Evolution Algorithm (MDEI) which uses interval analysis in three phases. In the first phase of MDEI, global minimum is roughly estimated with the application of interval arithmetic, which provides the upper and lower bounds of the objective function. In the second phase of MDEI we use an interval branch and bound algorithm, which we describe here along with cut off test and monotonicity test. It produces boxes with relatively small widths and gives upper and lower bounds of the objective function. The global minimizer exists with certainty in one of these boxes (Hansen, 1992). The above piece of information is used to initialize the population. In the third phase, MDEI algorithm modifies the new population, which is obtained after mutation, crossover and selection operations, by using a subset of new population. Then it constructs a shrinking box (Vrahatis, 1997) for this subset using interval arithmetic; and as the algorithm proceeds the width of shrinking box gets reduced. Finally the algorithm terminates with specified tolerance on the width of the shrinking box. As the formation of shrinking box depends on the number of individuals in the subset, it acts as a convergence check for the algorithm. Now we describe here the proposed algorithm in detail along with the associated algorithms.

### 3.1. Cut off test (Hansen, 1992)

The cut off test is considered as an accelerating device in the process of interval subdivision optimization, which uses the inclusion function $F(\mathbf{x})$ and the upper bound $\overline{f}$ for global minimum $f^*$. If the minimum value of $F(\mathbf{x})$ is greater than $\overline{f}$ then the box $\mathbf{x}$ is deleted. Let $\mathbf{y}$ be any subbox such that $\mathbf{y} \subseteq \mathbf{x}$. Let $\mathbf{y}$ be bisected to produce two subboxes $\mathbf{y}_1$ and $\mathbf{y}_2$. The cut off test is performed on $F(\mathbf{y}_1)$ and $F(\mathbf{y}_2)$, and returns the subboxes which are not discarded. The algorithm for cut off test is as described below.

**Algorithm Cut_ off_ test** : =Cut_off_test ($F(\mathbf{y}_1)$, $F(\mathbf{y}_2)$, $\overline{f}$, $\mathbf{y}_1$, $\mathbf{y}_2$)

Inputs : Inclusion functions $F(\mathbf{y}_1)$ and $F(\mathbf{y}_2)$, upper bound $\overline{f}$ for global minimum $f^*$, subboxes $\mathbf{y}_1$ and $\mathbf{y}_2$

Outputs : The subboxes which are not discarded.

BEGIN algorithm

1. {Execute for each subbox}
   if $\min F(\mathbf{y}_i) > \overline{f}$ then discard the subbox.
2. {Return}
   Output the subboxes which are not discarded.

END algorithm.

### 3.2. Monotonicity test (Hansen, 1992)

The monotonicity test determines whether $f(\mathbf{x})$ is strictly monotone over some subbox $\mathbf{y} \subseteq \mathbf{x}$ in some variable. Let $\nabla F(\mathbf{y})$ be the inclusion function of gradient of $f(\mathbf{y})$. If $0 \notin \nabla F(\mathbf{y})$ then the sub-box $\mathbf{y}$ is discarded. The monotonicity test is performed on the subboxes returned by cut off test. In this way it avoids unnecessary subdivisions and accelerates the interval subdivision algorithm.

**Algorithm Monotonicity_ test** : =Monotonicity_test ( $\nabla F(\mathbf{y})$, $\mathbf{y}$)
Inputs : The inclusion function $\nabla F(\mathbf{y})$ of gradient of $f(\mathbf{y})$, the subbox $\mathbf{y}$
Outputs : The subbox if not discarded.
BEGIN algorithm

1. {Execute for each subbox}
   if $0 \notin \nabla F(\mathbf{x})$ then discard the subbox.
2. {Return}
   Output the subbox if not discarded.

END algorithm.

### 3.3. Interval Subdivision Algorithm (Hansen, 1992)

The interval subdivision optimization method uses branch-and-bound algorithm as main deterministic algorithm and applies accelerating techniques like cut off test and monotonicity test. After sequential subdivision of the initial box $\mathbf{x}$ it produces a list of boxes of width $\varepsilon$ where the global minimizer is sure to lie. Also an interval $F^*$ containing the initial bounds for global minimum $f^*$is obtained (Hansen, 1992). Interval Subdivision algorithm is as follows.

**Algorithm Interval subdivision** [$\pounds$,$F^*$]:= Interval_ subdivision ($\mathbf{x}, f, \varepsilon, n,$ )

Inputs : The initial box $\mathbf{x}$ of $n$ dimensions, the function $f$, the maximum diameter $\varepsilon$ of the box for population initialization.

Outputs : The list of boxes $\pounds$, the interval $F^*$ containing the initial bounds for global minimum $f^*$.

BEGIN Algorithm

1. {Initialization}
   Set $\mathbf{y} = \mathbf{x}$ , $y = min\,(F(\mathbf{x}))$, $\overline{f} = maxF(\mathbf{x})$ as the upper bound for $f^*$, the working list $W = (\mathbf{y}, y)$ and the candidate list $\pounds = \{\}$.
2. **while** the list $W$ is not empty **do**
   - {Select $k$ direction for subdivision}
     $k = max\,\{w(\mathbf{y}_1), w(\mathbf{y}_2), ...w(\mathbf{y}_n)\}$
   - {Bisection}
     Bisect $\mathbf{y}$ normal to the direction $k$ obtaining two sub-boxes $\mathbf{d}_1$ and $\mathbf{d}_2$ such that $\mathbf{y} := \mathbf{d}_1 \cup \mathbf{d}_2$.
   - {Inclusion function formation}
     Evaluate $F(\mathbf{d}_1)$ and $F(\mathbf{d}_2)$and set $d_i = \min F(\mathbf{d}_i)$ for $i = 1, 2$ .
   - {Up dation of upper bound}
     $\overline{f} = \min\left\{\overline{f}, \max(F(\mathbf{d}_1)), \max(F(\mathbf{d}_2))\right\}$.

- {Up dation of working list $W$}
  Remove $(\mathbf{d}, d)$ from the working list $W$.
- {Perform Cut off test}
  for $i = 1, 2$ discard the pair $(\mathbf{d}_i, d_i)$ if $d_i > \overline{f}$.
- {Perform Monotonocity test}
  for $i = 1, 2$ discard the pair $(\mathbf{d}_i, d_i)$ if $0 \notin \nabla F_j(\mathbf{d}_i)$ for any $j \in \{1, 2, ..., n\}$
- {Insert the member to the list}
  if $w(F(\mathbf{d}_i)) < \varepsilon$ then insert the pair $(\mathbf{d}_i, d_i)$ to the candidate list $\pounds$.
  else insert the pair $(\mathbf{d}_i, d_i)$ to the working list $W$. The insertion is done in such a way that the second members $d_i$ of all pairs do not decrease.
- {Select the interval for subdivision}
  Set the first element of $W$ as $(\mathbf{y}, y)$

3. {Return}
   Output the list $\pounds$ and interval $F^*$

END Algorithm.

### 3.4. Initialization

#### 3.4.1. *Population Initialization*

Firstly we define the population size $NP$. The midpoints of the boxes in list $\pounds$ are taken as member of the initial population. In this way the number of individuals is equal to number of above boxes. If the number of boxes is greater than $NP$, then $NP$ is replaced by number of boxes. If it is smaller, the population is increased by taking sequentially a box from list $\pounds$, randomly selecting a point within it and adding this point to the population. This procedure is repeated until the number of individuals reaches the value $NP$.

#### 3.4.2. *Initialization of upper and lower bounds*

The MDEI algorithm uses the interval $F^*$ to initialize the upper bound $\overline{F^*}$ and lower bound $\underline{F^*}$ for global minimum $f^*$.

$$\underline{F^*} = infimum(F^*) \tag{13}$$

$$\overline{F^*} = supremum(F^*) \tag{14}$$

### 3.5. Population Modification

The population obtained after mutation, crossover and selection operations is modified for next generation using the subset $\mathbf{S}$ of population. This subset consists of $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$, where $k \leq NP$, which have the function values between the current upper bound $\overline{F^*}$ and lower bound $\underline{F^*}$. As the

number of individuals in **S** is smaller than the population size $NP$, the population is increased by taking sequentially a randomly selected individuals within **S** and adding this to the population. This procedure is repeated till the number of individuals reaches the value $NP$.

### 3.6. Termination criterion

The algorithm constructs a shrinking box (Vrahatis, 1997), denoted by $\mathbf{x}_s$ when the number of individuals $k$ in subset **S** exceeds a shrinking constant $r$ which is a user defined number and $r < NP$. The shrinking box is a interval vector, which contains the subset **S** of $n$ dimensional individuals. When a shrinking box is constructed, range estimation of $f$ over $\mathbf{x}_s$ is obtained using interval arithmetic. In this way, new bounds $\underline{F_s}$ and $\overline{F_s}$ are obtained. Thus, the current bounds of the global minimum are updated as follows

$$\overline{F}^* = \min\left\{\overline{F^*}, \overline{F_s}\right\} \tag{15}$$

$$\underline{F^*} = \max\left\{\underline{F^*}, \underline{F_s}\right\} \tag{16}$$

The algorithm makes the upper and lower bounds of $f^*$ sharper at each generation and terminates if $w(\mathbf{x}_s)$ and $w\left(F^*\right)$ are less than their respective tolerances $\varepsilon_x$ and $\varepsilon_f$.

As $w(\mathbf{x}_s)$ tends to zero the individuals are accumulated to a point as global minimizers. If $w\left(F^*\right)$ tends to zero, then the infimum of this interval is the global minimum $f^*$.

### 3.7. MDEI Algorithm

Algorithm MDEI [$f^*$,$\mathbf{x}^*$]:=MDEI($\mathbf{x}, n, r,\ f, \varepsilon, \mu,\ CR, NP,\ \varepsilon_x,\ \varepsilon_f$ )

Inputs : The initial box $\mathbf{x}$ of $n$ dimensions, shrinking constant $r$, the objective function $f$, the maximum diameter $\varepsilon$ of the box for population initialization, scaling factor $\mu$, the crossover constant $CR$, the population size $NP$, $\varepsilon_x$ tolerance for $x^*$and $\varepsilon_f$ tolerance for $f^*$

Outputs : global minimum $f^*$and minimizer $\mathbf{x}^*$.

BEGIN Algorithm

1. Apply interval subdivision algorithm.
2. {Initialization}

   Initialize the population with population size $NP$

   Initialize the upper and lower bounds
3. Evaluate each individual in the population.
4. **while** the termination criterion not reached **do**

   – Mutation
   – Crossover
   – Selection

- Modify the population.
- Update the upper and lower bounds.
- Select individuals for the next population.
- Evaluate the new individuals.

5. {Return}
   Output $f^*$and $\mathbf{x}^*$

END Algorithm

## 4. EELD Problem Formulation

The basic objective of economic emission load dispatch (EELD) of electric power generation is to obtain the optimal amount of generated power for the generating unit so as to meet the load demand at minimum operating fuel cost, satisfying all unit and system equality and inequality constraints.

Due to growing concern over the increase in the level of pollution, there arises a need to reduce the atmospheric emission. Thus the EELD problem can be formulated as a multi objective optimization problem, in which the emission, in addition to the fuel cost objective, is to be minimized. Here we have reduced the multi-objective problem to a single objective one by using different weighting factors (Abido, 1970) and (Slimani, 2007) .

### 4.1. Objectives

#### 4.1.1. *Fuel cost*

The most commonly used objective in the EELD problem formulation is the minimization of the total operation cost of the fuel consumed for producing electric power within a schedule time interval (one hour). The individual costs of each generating unit are assumed to be function, only of real power generation and are represented by quadratic curves. The total fuel cost $C(\mathbf{P})$ for the entire power system can then be expressed as the sum of the quadratic cost model of each thermal generating unit.

$$C(\mathbf{P}) = \sum_{i=1}^{N} a_i + b_i P_i + c_i P_i^2 \tag{17}$$

where $N$ represents the number of on-line generating units, $a_i, b_i$ and $c_i$ are the cost coefficients of the $i^{th}$generator and $P_i$ is the real power output of the $i^{th}$generator. $\mathbf{P}$ is the vector of real power outputs of generators and is defined as $\mathbf{P} = [P_1, P_2, ..., P_N]^T$.

#### 4.1.2. *Emission*

The emission control cost results from the requirement for power utilities to reduce their pollutant levels below the annual emission allowances assigned for the affected fossil units. The most important emissions considered in the power generation industry due to their effects on the environment are

sulphur oxides ($SO_x$)and nitrogen oxides ($NO_x$). To carry out an economic emission load dispatch, these emissions must be modeled through functions that relate emissions with power production for each unit. The total emission $E(\mathbf{P})$ can be expressed in a linear equation as the sum of all the pollutants resulting from each generator.

$$E(\mathbf{P}) = \sum_{i=1}^{N} \alpha_i + \beta_i P_i + \gamma_i P_i^2 + d_i \exp(e_i P_i) \tag{18}$$

where, $\alpha_i$, $\beta_i$, $\gamma_i$, $d_i$, and $e_i$ are emission coefficients of the $i^{th}$ generating unit emission characteristic.

### 4.2. Constraints

#### 4.2.1. *Power balance constraint*

The total electric power generation $\sum_{i=1}^{N} P_i$ must cover the total power demand $P_D$ and the real power loss $P_{loss}$ in the transmission line.

$$\sum_{i=1}^{N} P_i = P_D + P_{loss} \tag{19}$$

If the test system is considered to be lossless, the total electric power generation $\sum_{i=1}^{N} P_i$ must equal the power demand $P_D$. Thus,

$$\sum_{i=1}^{N} P_i - P_D = 0 \tag{20}$$

#### 4.2.2. *Generation capacity constraint*

Each generating unit is constrained by its lower and upper limits of real power output to ensure stable operation.

$$P_i^{\min} \leq P_i \leq P_i^{\max}, i = 1, 2, ..., N \tag{21}$$

where $P_i^{\min}$ and $P_i^{\max}$ are the minimum and maximum real power output of $i^{th}$ generator, respectively.

### 4.3. Total objective

In the EELD problem, the aim is to minimize the fuel cost as well as the emission. Hence, the multi-objective function is reduced to a single objective function $F(\mathbf{P})$ by assigning proper weights to fuel cost and emission, and is given by

$$F(\mathbf{P}) = \delta C(\mathbf{P}) + (1 - \delta) kE(\mathbf{P}) \tag{22}$$

where $k$ is emission control cost factor. Also $\delta$ is a weighting constant in the range $[0, 1]$ used as a weight factor for fuel cost and emission.

Table I. Fuel cost and emission coefficients for IEEE 30-bus 6-generator system.

| Generating. unit | $a_i$ | $b_i$ | $c_i$ | $\alpha_i$ | $\beta_i$ | $\gamma_i$ | $d_i$ | $e_i$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 200 | 100 | 4.091 | −5.554 | 6.490 | $2.0E-4$ | 2.857 |
| 2 | 10 | 150 | 120 | 2.543 | −6.047 | 5.638 | $5.0E-4$ | 3.333 |
| 3 | 20 | 180 | 40 | 4.258 | −5.094 | 4.586 | $1.0E-6$ | 8.000 |
| 4 | 10 | 100 | 60 | 5.326 | −3.550 | 3.380 | $2.0E-3$ | 2.000 |
| 5 | 20 | 180 | 40 | 4.258 | −5.094 | 4.586 | $1.0E-6$ | 8.000 |
| 6 | 10 | 150 | 100 | 6.131 | −5.555 | 5.151 | $1.0E-5$ | 6.667 |

### 4.4. Formulation

Aggregating the objectives and constraints, the problem can be mathematically formulated as

$$minimize F(\mathbf{P}) \tag{23}$$

subject to

$$\begin{aligned} g(\mathbf{P}) &= 0 \\ h(\mathbf{P}) &\leq 0 \end{aligned} \tag{24}$$

where $g$ and $h$ are the power balance equality and generation capacity inequality constraints respectively.

## 5. Experimental results

In this paper, the proposed algorithm MDEI using DE and intervals, has been developed using MATLAB 6.1. We investigate the effectiveness of the proposed approach over the conventional DE algorithm by considering the standard IEEE 30-bus 6-generator as a test system. The values of fuel cost coefficients and emission coefficients are given in Table I. The system is considered lossless. Therefore, the problem constraints are the power balance constraint without $P_{loss}$ and the generation capacity constraint. The generation capacity in p.u. for all generating units are taken to be same and is given as

$$P_i = [0.5, 1.5], i = 1, 2, ..., 6 \tag{25}$$

The remaining data required for the EELD problem is taken as

$$k = 30.0738 \tag{26}$$

$$P_D = 2.834 p.u \tag{27}$$

We have considered here two cases depending on the value of weighting constant $\delta$.

Table II. Comparison of the proposed MDEI algorithm and the conventional DE algorithm for Case 1.

| *Algorithm* | *BV* | *NS* | *ENES* | *MNE* |
|---|---|---|---|---|
| *DE* | 600.23 | - | - | 9868 |
| *MDEI* | 600.11 | 20 | 5950 | 6190 |

– Case 1 : $\delta = 0$; when the minimization of the objective function is done without the emission objective.

– Case 2 : $\delta = 1$; when the minimization of the objective function is done without the fuel cost objective.

For a fair comparison we executed 20 test runs of both the algorithms on the EELD problem. Let $NS$ be the number of runs out of the 20 runs that succeeded in finding the global minimum function value $F^*$ within the tolerance $\varepsilon_f = 10^{-3}$. Let $NE$ be the total number of function evaluations for which $F^*$ were obtained during the 20 runs. Then, the Expected Number of Evaluations per Success, $ENES$ represents the mean number of function evaluation needed in order that the algorithms reach the termination criterion and is computed as $ENES = NE/NS$. If the global minima can not be achieved, then $ENES$ is not defined. $MNE$ is the maximum number of real function evaluations of a particular run, out of total twenty runs of the algorithm. To calculate the total effort we adopt the procedure as given in [11]. Assuming that one interval evaluation is equivalent to two real number evaluation, the total effort $TE = 2 * (IFE + IGE) + MNE$, where $IFE$ is the total number of interval function evaluations, $IGE$ is the total number of interval gradient evaluations.

For all the optimization runs of both the algorithms, the population size $NP$ was set as 50, mutation scaling factor $\mu$ as 0.95 and crossover constant $CR$ as 0.8. The tolerance for approximating the minimizer $\mathbf{P}^*$ is taken as $\varepsilon_x = 10^{-7}$.

**Case 1** : With $\delta = 0$, the optimization of objective problem provides the best fuel cost solution. The interval subdivision algorithm in MDEI gave the initial bounds for $F^*$ as $[175.4594, 1950.1]$ in 755 boxes, each of width 0.7 with $2 * (IFE + IGE) = 3678$ evaluations. This large amount of computational effort is due to high dimensionality of the problem as well as to the wide range of search space. The complete MDEI algorithm gave $MNE = 2512$, making $TE = 6910$. For DE algorithm $TE = MNE$. $BV$ is the best global minima of the objective function which each of the algorithm has obtained. The results are exhibited in Table II. A hyphen in the table indicates that corresponding value of the field could not be evaluated.

From Table II we observe that MDEI is the only algorithm that has found a better global minimum with certainty. We also observe that MDEI has succeeded in all the runs due to which $ENES$ could be computed as to be 5950. From this it is evident that an efficient termination criterion has been developed, which is verified in all runs. For each run the maximum number of real function evaluations has been less than or equal to 6190 which is quite less than that of DE algorithm. But DE has not succeeded even in a single run, due to which $ENES$ could not

Table III. Comparison of the proposed MDEI algorithm and the conventional DE algorithm for Case 2.

| *Algorithm* | *BV* | *NS* | *ENES* | *MNE* |
|---|---|---|---|---|
| *DE* | 572.1115 | - | - | 11278 |
| *MDEI* | 572.1112 | 20 | 6578 | 7600 |

Table IV. Optimum power outputs of each generator in *p.u.*in both the cases.

| Generating. unit | Case 1 *p.u.* power | Case 2 *p.u.* power |
|---|---|---|
| $P_1$ | 0.11 | 0.3904 |
| $P_2$ | 0.30 | 0.4932 |
| $P_3$ | 0.524 | 0.5025 |
| $P_4$ | 1.016 | 0.4533 |
| $P_5$ | 0.524 | 0.5024 |
| $P_6$ | 0.360 | 0.4921 |

be computed and it consumes all the trials and terminates without any guarantee that the global minimum has been found. The output power in p.u. of all generators obtained from MDEI are given in Table IV.

**Case 2** : With $\delta = 1$, the optimization of objective problem provides the best emission solution. The interval subdivision algorithm in MDEI gave the initial bounds for $F^*$ as $[237.9146, 2036.7]$ in 755 boxes, each of width 0.7 with $2 * (IFE + IGE) = 3678$ evaluations. The complete MDEI algorithm gave $TE = 7600$. For DE algorithm $TE = 11278$ . $BV$ is the best global minima of the objective function which each of the algorithm has obtained. The results are exhibited in Table III.

From Table III we observe that for each run the maximum number of real function evaluations using MDEI is at most 7600 and $ENES$ is 6578. We also observe that DE algorithm has not succeeded even in a single run as shown by hyphen, thus $ENES$ could not be defined for it. The results exhibited in Table III clearly shows that MDEI has succeeded in all the rums and found the global minima with certainty. The output power in p.u. of all generators obtained from MDEI are given in Table IV.

## 6. Conclusions

In this contribution a modified Differential Evolution Algorithm using intervals MDEI was presented. Randomly formed initial population and poorly defined termination criterion made the

conventional DE computationally expensive and also the global minimum was not guaranteed. The above drawbacks were overcome by the inclusion of interval arithmetic in conventional DE. The algorithm used the branch-and-bound principal to obtain small regions where candidate solutions lie. In this way, a highly performing initial population was formed. Then the upper and lower bounds of the objective function were updated at each generation. Finally, the algorithm was able to compute the global minimum with certainty.

The proposed algorithm became more effective when a new termination criterion based on the concept of shrinking box was used. The extra computational effort was avoided by selecting the next generation from the subset of current population, which performed with in the upper and lower bounds of the function.

We applied the proposed algorithm to a EELD problem of IEEE 30 bus 6 generators system and compared the results with that obtained from conventional DE. We demonstrated that MDEI exhibited high performance with a high dimensional problem like EELD. Hence, we conclude that MDEI outperforms conventional DE.

## References


Sotiropoulos, D. G. ,E. C. Stavropolous, and M. N. Vrahatis. A New Hybrid Genetic Algorithm for Global Optimization. *Nonlinear Analysis*, 33(7):4529–4538, 1997

Storn, R. Differential Evolution, A Simple and Efficient Heuristic Strategy for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997

Dhillon, J. S. ,S. C. Parti, and D. P. Kothari. Stochastic Economic Emission Load Dispatch. *Electric Power Systems Research*, 26:186–197, 1993

Slimani, L. ,T. Bouktir. Economic Power Dispatch of Power System with Pollution Control using Multiobjective Ant Colony Optimization. *International Journal of Computational Intelligence Research*, 3(2):145–153, 2007.

Goldberg, D. E . *Genetic Algorithms for search, Optimization, and Machine learning,Reading.* Addison-Wesley Publishing Company, 1989.

Lampinen, J. Global Optimization by Differential Evolution. Technical Report, Lappeenranta University of Technology, Lappeenranta, Finland, 1999.

Koridak, L. A. ,M. Rahli, and M. Younes. Hybrid Optimization of the Emission and Economic Dispatch by the Genetic Algorithm. *Leonardo Journal of Sciences*, 14:193–203, 2008.

Moore, R. E. *Interval analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

Hansen, E. *Global optimization using interval analysis.* Dekker, New York, 1992.

Zielinski, K.,D. Peters, and R.Laur Stopping Criteria for Single-Objective Optimization. In *In Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2005

Abido, M. A. Abido. *IEEE Transactions on Enolutionary computation*, 10(3):145–153, 2006.