A Differential Evolution Based Memetic Algorithm for Workload Optimization in Power Generation Plants

Ankush Mandal, Swagatam Das
Department of Electronics and Telecomm. Engg.,
Jadavpur University, Kolkata 700032, India
E-mails: ankushmandal19@gmail,
swagatamdas19@yahoo.co.in

Ajith Abraham
[2]Machine Intelligence Research Labs (MIR Labs)
Scientific Network for Innovation and Research
Excellence, WA, USA
[3]IT For Innovations, EU Center of Excellence,
VSB - Technical University of Ostrava, Czech Republic
E-mail: ajith.abraham@ieee.org

***Abstract*—Work load optimization in power generation plants is of practical importance in carbon constrained power industry. The main objective of the coal-fired power generation workload optimization is to minimize fuel consumption while maintaining the desired output and to maintain NO_x emission within the environmental license limit. In this article, we represent an efficient Memetic Algorithm (MA) with a constraint handling method for the power generation loading optimization. This MA is developed by combining a competitive variant of Deferential Evolution (DE) and Simplex method. The proposed approach incorporates the constraint handling method to modify the selection rule which guides the search process in better direction. The simulation results based on a coal-fired power plant clearly indicate that our proposed method is very effective and it shows great computational efficiency in power generation workload optimization.**

Keywords-Memetic Algorithm; Differential Evolution; Local Search; Power Generation Workload Optimization

I. INTRODUCTION

Usually there are several generating units in a power generation plant. How to make the best use of the units is a major objective of any company. Increased pressures from environmental regulations, rising fuel costs, and green house gas emissions demand power generators to be more efficient. For a typical power utility with a number of units, the unit thermal efficiencies alter all the time. A unit's thermal efficiency is determined by many factors such as design, construction, fuel and ambient conditions, level of maintenance and operation skills etc. For a large power company with different kinds of units, optimizing workload distribution is of practical importance in terms of fuel saving and minimizing environmental harm.

Generally, a power generation company has a *m*-year (or *m*-month) overhaul system. It means each time, a unit is through a major overhaul in turn and every *m* years (or months) the plant completes an overhaul cycle. The most recently overhauled unit will have highest thermal efficiency and the one close to an overhaul will have lowest thermal efficiency. Units with higher thermal efficiency will consume less fuel and cause less environmental harm while units with lower thermal efficiency will consume more fuel and lead to higher environmental harm. In the normal operation range, as load increases, unit thermal efficiency increases (or heat rate decreases). The thermal efficiency for each unit also depends on what kind of problems it developed, what modifications it went through, and what operation mode a unit is operating under (such as mill pattern etc). The optimized loading can be achieved based on the units' thermal efficiency and NO_x emission characteristics for a given plant condition.

We have two objectives in the power generation loading optimization problem. One is to minimize the total heat consumption (fuel consumption) and another one is to maintain the total NO_x emission within the environmental license limit. However, the second objective is basically a constraint since the NO_x emission level must be within the license limit whatever the situation may be. In practice, it is desirable that the unit with higher thermal efficiency (lower heat rate) receives higher workload and the unit with lower thermal efficiency (higher heat rate) receives lower workload.

In recent years, Evolutionary Algorithms (EAs) [1, 2] have been applied in many real world optimization problems [3-5]. EAs with proper enhancement are good choice for real world optimization problems because they are inspired by natural phenomena where the interfacing is with real world.

Differential Evolution (DE) algorithm [1] belongs to EAs and now-a-days it is considered one of the most powerful tools in EAs. It can be interpreted as discrete dynamical system governing the movements of a set of vectors in the search space. Behavior of the individuals greatly influence the progress of the search process and therefore on the convergence of the algorithm. However, EAs are good for global optimization where the exploration of the entire search space is required within relatively small no of iteration. But they are not good for producing precise solutions.

Local Search (LS) algorithms [6, 7] are used to explore the nearby region around the current solutions with high intensity. So, by using a LS method, highly accurate solutions can be obtained. EAs hybridized with LS method are commonly called Memetic Algorithms (MAs) and these MAs [8, 9] are proven to be more efficient than the EAs themselves. The reason behind this is the combination of global exploration and local exploitation.

978-1-4577-2150-2

In this article, we propose a MA, denoted by cDESimplex, which integrates a constraint handling method to obtain a valid optimized power generation workload distribution. In our proposed algorithm, a competitive variant of DE is used for global exploration and Simplex method [7] is used as the local search process. For the DE algorithm, we have developed a hybrid mutation strategy by hybridizing a modified "DE/current-to-best/2" mutation strategy with a modified "DE/rand/1"mutation strategy. We have discussed the mutation strategy later in sufficient details.

The rest of this paper is organized as follows: section II describes classical DE. Section III gives details about the problem formulation. A detailed description of the proposed cDESimplex algorithm is given in section IV. Section V describes the parameter settings. In section VI, experimental results are represented and analyzed thoroughly. Finally, section VII concludes the paper.

II. CLASSICAL DE

Differential evolution (DE) algorithm, proposed by Storn and Price [1] is a simple but effective population-based stochastic search technique for solving global optimization problems.

1) Initialization of the Population: Let $S \subset \Re^D$ be the D dimensional search space of the problem under construction. The D.E. evolves a population of NP D-dimensional individual vectors, $\vec{X}_i = \{x_i^1, x_i^2, ..., x_i^D\}$ $i=1,2,...NP$ from one generation to another. The initial population should ideally cover the entire search space by randomly distributing each parameter of an individual vector between the prescribed upper and lower bounds x_j^u and x_j^l, $j \in [1, D]$ respectively.

For every generation G, the D.E. employs mutation and crossover operation corresponding to every target vector $\vec{X}_{i,G}$ and produces the trial vector, $\vec{U}_{i,G}$

2) Mutation operation: For every target vector, $\vec{X}_{i,G}$, in any generation G, a mutant_vector $\vec{V}_{i,G}$ is generated according to a certain mutation scheme. The most common mutation policies are:

a) DE/rand/1/bin: $\vec{V}_{i,G} = \vec{X}_{r1,G} + F \cdot (\vec{X}_{r1,G} - \vec{X}_{r3,G})$ (1)

b) DE/best/1/bin: $\vec{V}_{i,G} = \vec{X}_{best,G} + F \cdot (\vec{X}_{r1,G} - \vec{X}_{r2,G})$ (2)

c) DE/current-to-best/2/bin:

$\vec{V}_{i,G} = \vec{X}_{i,G} + F_1 \cdot (\vec{X}_{best,G} - \vec{X}_{i,G}) + F2 \cdot (\vec{X}_{r1,G} - \vec{X}_{r2,G})$ (3)

Where r1, r2, and r3 are random and mutually exclusive integers generated in the range [1, NP], which should also be different from the trial vector's current index ***i***. F is a factor for scaling differential vectors and $\vec{X}_{best,G}$ is the individual vector with best fitness value in the population at generation G.

3) Crossover operation: This operation involves binary crossover between the target vector $\vec{X}_{i,G}$ and the mutant vector $\vec{V}_{i,G}$ produced in the previous step which produces the trial vector $\vec{U}_{i,G} = \{u_{i,1,G}, u_{i,2,G}, ...u_{i,D,G}\}$. The crossover operation is done as follows.

$$u_{\mathrm{i,j,G}} = \begin{cases} v_{i,j,G} & \text{if } rand(0,1) \le CR \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases} \quad (4)$$

where CR is a user-specified crossover constant in the range [0, 1) and j_{rand} is a randomly chosen integer in the range [1, D] to ensure that the trial vector $\vec{U}_{i,G}$ will differ from its corresponding target vector $\vec{X}_{i,G}$ by at least one component.

4) Selection Operation: If the values of some parameters of a newly generated trial vector exceed the corresponding upper and lower bounds, we randomly and uniformly reinitialize it within the search range. Then the fitness values of all trial vectors are evaluated. After that, a selection operation is performed. The fitness value of each trial vector $f(\vec{U}_{i,G})$ is compared to that of its corresponding target vector $f(\vec{X}_{i,G})$ in the current population and the population for the $(G+1)$ generation is formed as follows: (for a maximization problem)

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \ge f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \text{otherwise} \end{cases} \quad (5)$$

where $f(.)$ is the objective function.

The above 3 steps are repeated until some stopping criteria is satisfied.

III. PROBLEM FORMULATION

In this section, we have introduced the necessary terms and notations used in the power generation loading optimization.

1) Total load demand, denoted as L_{total} (MW), is the total power demand by the market.

2) Unit load, denoted as x_i (MW), the workload allocated to unit i.

3) Environmental license limit of NO_x emission, denoted as $N_{\max}$ (g/ m^3), is the maximum emission allowed for each unit;

4) Unit heat rate, denoted as f_i (KJ / KW. h), is the heat consumption for generating per unit (KW. h) electricity. For a given condition, the heat rate is a function of unit load and can be expressed in a polynomial form, which is obtained from field testing and unit modeling. The general expression for the heat rate function for unit i is:

$$f_i(x_i) = a_{i,0} + a_{i,1}x_i + ...+ a_{i,(k-1)}x_i^{(k-1)} + a_{i,k}x_i^k \quad (6)$$

where these a_i are the coefficients of the polynomial, k is the order of polynomial function.

5) Heat consumption, denoted as H (MJ / h), is the unit heat consumption per hour at a given load. Heat consumption by the ith unit can be expressed as:

$$H_i = x_i f_i(x_i) \quad (7)$$

6) Unit NO_x emission level, denoted as q (g/m^3), is the amount of emission for a given load. However, each unit has its own emission curve which is generally a linear function in the normal operation range. It is obtained from the field testing and unit modeling. In general form it can be expressed as:

$$q_i(x_i) = b_{i,o} + b_{i,1} x_i \quad (8)$$

where b_i are the coefficients.

The main objective of the workload optimization process is to determine the optimal unit load so as to minimize the total heat consumption. The total heat consumption is the sum of all units' heat consumption, which can be expressed as the following:

$$F(X) = \sum_{i=1}^{D} H_i = \sum_{i=1}^{D} x_i f(x_i) \quad (9)$$

where *D* is the number of units.

For the optimization process, there are several constraints need to be satisfied.

1) The total load demand must be achieved at a given time. The constraint can be expressed as:

$$\sum_{i=1}^{D} x_i = L_{total} \quad (10)$$

Considering the data type that will be implemented in double precision, this constraint can be modified as:

$$\left| \sum_{i=1}^{D} x_i - L_{total} \right| < \varepsilon \quad (11)$$

where ε is a minimum error criterion for equality constraint.

2) The NO_x gas emission from each unit has to be restricted within the license limit $N_{\max}$. This constraint can be expresses as:

$$q_i(x_i) - N_{\max} \le 0 \ (for\ i=1, 2, \ldots, D) \quad (12)$$

3) There are also unit capacity constraints. For stable operation, the workload for each unit must be restricted within its lower and upper limits. This is the range where a unit load can be readily adjusted without excessive human intervention, for example, a unit is operating between 60% to 100% load without the need of mill change. Let $L_{i,\min}$ and $L_{i,\max}$ represent the lowest and highest limits of workload for unit number *i* respectively, the constraint then can be expressed as:

$$L_{i,\min} \le x_i \le L_{i,\max} \ (for\ i=1, 2, \ldots, D) \quad (13)$$

The unit capacity constraints can be modeled as the boundary constraints in the optimization.

Now, the whole optimization problem can be summarized as:

$$\text{Minimize: } F(X) = \sum_{i=1}^{D} x_i f_i(x_i) \quad (14)$$

$$\text{Subjected to: } g(X) = \left| \sum_{i=1}^{D} x_i - L_{total} \right| - \varepsilon < 0 \quad (15)$$

$$r_i(x_i) = q_i(x_i) - N_{\max} \le 0 \ (for\ i=1, 2, \ldots, D) \quad (16)$$

where

$$f_i(x_i) = a_{i,0} + a_{i,1} x_i + \ldots\ldots + a_{i,(k-1)} x_i^{(k-1)} + a_{i,k} x_i^k \quad (17)$$

$$L_{i,\min} \le x_i \le L_{i,\max} \ (for\ i=1, 2, \ldots, D) \quad (18)$$

IV. Proposed Algorithm

In this section we have discussed the proposed cDESimplex algorithm in sufficient details. In this multi-population based algorithm, we have developed a competitive variant of DE which is accompanied by a local search method. Furthermore, this algorithm employs a hybrid mutation strategy for DE to enhance the searching ability and to circumvent stagnation of the population at any local optima.

A. The modified DE algorithm

1) Competitive variant of DE: For a heuristic search process, it is useful to exploit the neighborhood because it is similar to information exchange between the neighbors which leads to better solutions. So, here we have incorporated a competition between the neighbors. Also the success rate is measured at each generation which helps in determining the new individual generation process for the next generation. Actually, depending on the success rate, either the current individual or its nearest neighbor is used for mutant vector formation. If the corresponding trial vector is chosen for next generation then the corresponding success rate is increased by 1 and if it is not chosen then the corresponding success rate is decreased by 1. If both the success rates for current individual and its nearest neighbor are equal then the current individual is used. At the time of initialization, all the success rates were set to 0. Using this method, we can get far better solutions with less function evaluations. Also the population does not converge to any local minima too quickly because we set the competition with nearest neighbor. Here, the nearest neighbor is selected on the basis of Euclidean distance between the current individual and the other individuals in the corresponding subpopulation.

2) Hybrid mutation strategy: As mentioned earlier, depending on the success rate, either the current individual or the nearest neighbor of the current individual is used for the mutant vector generation process. Let the chosen individual be $\vec{X}_{c,G}$.

In DE, greedy strategies like DE/current-to-best/*n* and DE/best/*n* benefit from their fast convergence property by guiding the search process with the best solution so far discovered, thereby converging faster to that point. But, as a result of such fast convergence tendency, in many cases, the population may lose its diversity and global exploration abilities within a relatively small number of generations, thereafter getting trapped to some locally optimal point in the search space. Taking into consideration these facts and to overcome the limitations of fast but less reliable convergence, we have developed a hybrid mutation strategy.

For constructing the final mutant vector, two mutant vectors generated by two different mutation strategies are combined with a weight factor ω. This way we developed a hybrid

mutation strategy to prevent the algorithm from converging too quickly and at the same time exploring the whole search space to produce high quality results.

For the first mutation strategy, we have used a modified "DE/current-to-best/2" mutation strategy. For this modified mutation strategy, best individual of each subpopulation is stored in a memory archive; this memory archive is updated at each generation to store the new best individuals and delete the previous best individuals. During the mutation process, the nearest memory individual is used instead of the global best individual. The mutation process can be expressed as follows:

$$\vec{V}_{mut,1} = \vec{X}_{c,G} + F_{best} \cdot (\vec{X}_{m,G} - \vec{X}_{c,G}) + F \cdot (\vec{X}_{r1,G} - \vec{X}_{r2,G}) \quad (19)$$

where $\vec{X}_{m,G}$ is the nearest best individual as mentioned above. $\vec{X}_{r1,G}$ and $\vec{X}_{r2,G}$ are two distinct vectors randomly chosen from the subpopulation.

For the second mutation strategy, we have used "DE/current/1" mutation strategy. The mutation process can be expressed as follows: $\vec{V}_{mut,2} = \vec{X}_{c,G} + F \cdot (\vec{X}_{r1,G} - \vec{X}_{r2,G})$ (20)

where $\vec{X}_{r1,G}$ and $\vec{X}_{r2,G}$ are two distinct vectors randomly chosen from the subpopulation independently of first mutation process.

Now, the final mutant vector is a weighted sum of two above mentioned mutant vectors. If the weight factor for $\vec{V}_{mut,1}$ is ω then the final mutant vector is

$$\vec{V}_{mut} = \omega \cdot \vec{V}_{mut,1} + (1-\omega) \cdot \vec{V}_{mut,2} \quad (21)$$

B. Local Search

As mentioned earlier, we have used Simplex Method as the LS algorithm. The conventional simplex method [7] proposed by Nelder and Mead is a widely accepted search technique. A simplex in an D dimensional space is defined by a convex polyhedron consisting of D+1 vertices denoted as $x_1, x_2, ..., x_D, x_{D+1}$ of which the worst, the second worst, and the best point are respectively denoted as x_w, x_s and x_b. The centroid x_m of the remaining vertices except for x_w is defined by

$$x_m = \frac{(x_1 + x_2 + ... + x_D + x_{D+1} - x_w)}{D} \quad (22)$$

The search procedure of the simplex method involves four basic operations: reflection, expansion, compression, and contraction.

1) Reflection: The reflected point x_r of x_w across the centroid x_m is generated by

$$x_r = x_m + (x_m - x_w) \quad (23)$$

If x_r is worse than x_b but still not than x_s then x_w is replaced with x_r, else, one of the following alternative operations is performed:

2) Expansion: If x_r is not worse than x_b, a new point x_e is generated further along the reflection direction in the following way:

$$x_e = x_m + \alpha \cdot (x_m - x_w) \quad (24)$$

where $\alpha > 1$ is called the expansion coefficient. If x_e is better than x_b then x_w is replaced with x_e or else with x_r.

3) Compression: If x_r is worse than x_s, let x_k be the better one of x_r and x_w. A new point x_c close to x_m is generated by:

$$x_c = x_m + \beta \cdot (x_k - x_m) \quad (25)$$

where $0 < \beta < 1$ is called the compression coefficient. If x_c is not worse than x_k, x_w is replaced with x_c.

4) Contraction: If reflection, expansion, and compression cannot obtain a promising point (i.e., at least better than x_w) D new points are generated by

$$x_i = x_b + \frac{x_i - x_b}{2} \quad (26)$$

where $i = \{1,2,..., D+1\} - \{b\}$ and the corresponding points in the original simplex are replaced with D new ones. By repeatedly generating new points using one of the four basic operations, the simplex method finds its way downhill to converge toward an optimum. The simplex method does not need gradient; however, it heuristically uses the local information about the search landscape.

In our algorithm, during the local search process, the local search method is applied over all subpopulations separately. After every *Gen_de* (we set it to 1000) iterations we applied Local Search process to the populations. Each LS run is allotted *F_ls* (we set it to 600) function evaluations.

C. Modified Selection Rule

In order to incorporate the constraint handling method to our proposed algorithm, we modified the selection rule as follows:

> (*Let us assume we are applying selection rule to the individuals* X_1 *and* X_2.)
> ***If*** X_1 has constraint violation more than X_2 ***then*** X_2 is selected for the next generation.
> ***else if*** X_2 has constraint violation more than X_1 ***then*** X_1 is selected for the next generation.
> ***else*** both X_1 and X_2 satisfy the constraints ***then*** the one with less objective function value ($F(X)$) is chosen for the next generation.
> ***end if***

Constraint violation is measured by the following function:

$$\Phi(X) = g(X) + \sum_{i=1}^{D} r_i(x_i) \quad (27)$$

provided $g(X) > 0$ and $r_i(x_i) > 0$ (if for any $i \in \{1,2,....D\}$, $r_i(x_i) \le 0$ then $r_i(x_i)$ is set to 0 for that i and if $g(X) \le 0$ then $g(X)$ is set to 0)

Actually, the constraint must be followed at any condition. If a workload distribution does not follow the constraint then it is not a valid solution irrespective of how low its objective value is. So, it is wise to give the first preference to constraint

then to consider the objective function value for any individual in the populations of the search process.

V. Parameter Settings

A. Population Size

Population size (*NP*) was kept fixed at 60 throughout the search process. We divided the whole population into 6 subpopulations, each containing 10.

B. Scaling Factor

We In this algorithm, scaling factor for each dimension of the difference vector is generated randomly depending on the value of the difference vector along the corresponding dimension. F is generated independently for each dimension of the difference vector. Scaling factor generation can be explained as follows:

$$F_j^i = rand(0,1) \cdot e^{-\left|x_j^i\right| / \left|x_R^i\right|} \quad (28)$$

where $i \in$[1, D(Dimension of the search space)]. We are generating scaling factor for the i th dimension of the j th individual. x_j^i is the value of the difference vector along i th dimension, x_R^i is the search range along that dimension.

C. Weight factor for hybrid mutation strategy:

In Weight factor ω for the first mutation scheme in hybrid mutation strategy was set to 0.7.

D. The minimum error criterion for equality constraint:

The minimum error criterion for equality constraint (ε) was set to *1.0E-03*.

E. The NO_x emission license limit:

Environmental license limit of NO_x emission $N_{\max}$ is $1.3(g/m^3)$.

F. The feasibility tolerance:

The feasibility tolerance allowed δ = *1.0E-8*, that is, if a solution's total amount of constraint violation $\Phi \leq \delta$, the solution is considered feasible.

G. Maximum number of iterations:

Maximum number of iterations was set to 5000.

VI. Experimental Results

A. Simulation Environment

For the simulation process, we chose a local power plant which has four 360MW and a total generation capacity of 1440MW. It has a four-year overhaul system, i.e. each year, a unit is through a major overhaul in turn and every four year the plant completes an overhaul cycle. The boundary constraints $L_{i,\min}$ and $L_{i,\max}$ for each unit are 220 (MW) and 360 (MW). The total load output of the power station ranges from 4*220 = 880 (MW) as the minimum to 4*360 =1440 (MW) as the maximum. It would be better to simulate a series of output (a dynamic L_{total}) so that to allow the power plant to choose from the optimal results according to the market demand.

The heat rate functions and the NO_x emission functions for the four generator units are provided from a local power plant setting. The heat rate functions are in the polynomial format with the power of two. The NO_x emission functions are linear. The sample functions are listed in Table 1. These functions can be modified when the units' performances are changed.

Table 1. Unit Heat Rate and NO_x Emission Functions

Unit No.	Type	Function
1	Heat Rate	$f_1(x_1) = 0.0023x_1^2 - 3.7835x_1 + 9021.7$
	NO_x Emission	$q_1(x_1) = 0.0036x_1 - 0.1717$
2	Heat Rate	$f_2(x_2) = 0.0238x_2^2 - 9.7773x_2 + 9432.6$
	NO_x Emission	$q_2(x_2) = 0.0031x_2 - 0.0226$
3	Heat Rate	$f_3(x_3) = 0.0187x_3^2 - 5.3678x_3 + 10240.0$
	NO_x Emission	$q_3(x_3) = 0.0036x_3 - 0.1252$
4	Heat Rate	$f_4(x_4) = 0.012x_4^2 - 5.7450x_4 + 9231.7$
	NO_x Emission	$q_4(x_4) = 0.0039x_4 - 0.1706$

B. Result analysis and discussion

In Table II presents the simulation results to the whole range of the generation capacity. Based on the efficiency functions as listed in the Table I, the optimal workloads to the four generators have been found for each total output demand. As desired, the unit with higher thermal efficiency receives higher workload (in this case Unit 1) while the unit with lower thermal efficiency receives lower workload (in this case Unit 3) after the optimization process. In practice, when the total output load changes, the optimal load allocation can be found from these data. For the minimum (880MW) and maximum (1440MW) loading conditions, there is no gain from the optimization since no options for loading at both ends.

To prove that our proposed algorithm has much more computational efficiency than other state-of-the-art EAs, we have optimized the same problem by PSO [2] and DE [1]. An identical environment was maintained for all the algorithms. Table 3 clearly indicates that our proposed algorithm is much faster in terms of computational time than PSO and DE. The main reason is that our proposed algorithm efficiently uses

both the exploration and exploitation properties which help the search process to find the optimum faster. Moreover, as it uses a competition between the individuals in each population, the information exchange between the individuals within each population is greater than in other EAs. Also, the hybrid mutation strategy circumvents stagnation which in turn saves the search process from unnecessary function evaluations.

It is worth mentioning that a static environment is used for the current optimization process. That is, the objective function and constraints function are static for a specific case. However, in the real world applications, the output demand constraint can be time-varying. The objective function can be considered static or dynamic. It would be interesting to study methods to optimize such challenging problems in the dynamic environment.

TABLE 2. OPTIMIZED WORKLOAD DISTRIBUTION

L_{total} (MW)	Unit 1 (MW)	Unit 2 (MW)	Unit 3 (MW)	Unit 4 (MW)
880	220.0000	220.0000	220.0000	220.0000
900	237.1295	220.3828	220.0135	222.4742
950	274.2694	223.5023	220.5023	232.0854
1000	328.7876	226.9730	220.0032	226.2363
1050	359.4194	225.0253	221.5988	243.9572
1100	359.8675	227.8953	221.1147	291.1222
1150	359.8896	235.5483	221.1814	333.3800
1200	359.9929	269.2605	247.5110	323.2364
1250	359.9937	326.3054	221.1532	342.5464
1300	359.1863	339.7221	243.1138	357.9769
1350	359.3939	331.9112	299.7438	358.9509
1400	359.4781	353.5394	327.1062	359.9299
1440	360.0000	360.0000	360.0000	360.0000

TABLE 3. COMPARISON OF COMPUTATIONAL TIME REQUIREMENTS (BASED ON 20 INDEPENDENT RUNS)

CPU time spent	cDESimplex (ms)	DE (ms)	PSO (ms)
Minimum	28	79	126
Maximum	147	335	519
Average	61.75	182.17	252.33

VII. CONCLUSIONS

This article represents an efficient optimization method for determining the optimum workload distribution in power plants using the proposed cDESimplex algorithm. The simulation results obtained over a practical Power plant clearly indicates that the proposed method can perform an efficient search for optimal loading condition without violating the environmental license limit for NO_x emission. It is also clear from the results that cDESimplex algorithm can obtain higher quality solutions within much less computational time than DE or PSO.

Our future work will include a study over the power generation workload optimization with dynamic environment which more challenging in nature.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization,* vol. 11, 341-359, 1997.

[2] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in Proceedins of IEEE International Conference on Neural Networks. vol. 4 Perth, Australia: IEEE Service Centerr, 1995, pp. 1942-1948.

[3] R. Joshi and A.C. Sanderson, "Minimal representation multi-sensor fusion using differential evolution", *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 29, no. 1, pp. 63-76, 1999.

[4] M. A. Abido, "Multiobjective evolutionary algorithms for electric power dispatch problem", IEEE Transactions on Evolutionary Computation, vol. 10, 315 – 329, 2006.

[5] D. E. Goldberg, Genetic Algorithms for Search, Optimization, and Machine Learning: Addison-Wesley, 1989.

[6] F. J. Solis and R. J. Wets, "Minimization by random search techniques", mathematical Operations Research, 6:19–30, 1981.

[7] S. Smith, "The simplex method and evolutionary algorithms," *Proc. 1998 IEEE Int. Conf. Evolutionary Computation*, pp. 799–804, 1998.

[8] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization", in IEEE Congree on Evolutionary Computation, 2010.

[9] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, E. Ebrahimi, and C. Lucas, "Self-adaptive memetic algorithm: an adaptive conjugate gradient approach", IEEE Conference on Cybernetics and Intelligent Systems, 2004, 6 – 11.