American Journal of Electrical and Electronic Engineering, 2015, Vol. 3, No. 4, 100-106
Available online at http://pubs.sciepub.com/ajeee/3/4/3
© Science and Education Publishing
DOI:10.12691/ajeee-3-4-3

Smart Evolutionary Algorithm for Static Economic Load Dispatch Optimization in a Thermal Generating Station

Sunny Orike*, Vincent I. E. Anireh

Department of Electrical/Computer Engineering, Rivers State University of Science and Technology, Port Harcourt, Nigeria
*Corresponding author: orike.sunny@ust.edu.ng

Abstract The Economic Load Dispatch (ELD) problem is an optimization task with emphasis on how power generating companies (GENCOs) will be able to meet the power demands of the distribution companies (DISCOs) and electricity consumers, and at the same time minimize both under/over generation of electricity, and also minimize the operational costs of running the units in their various stations. This paper implemented a Smart Evolutionary Algorithm, which combines a standard Evolutionary Algorithm with a smart mutation operator that is applied to the Static ELD problem. It also investigated and analyzed three distinct variants of the smart mutation operator. The operator focused mutation on genes contributing mostly to cost of generation and penalty violations in the fitness function. Rather than using a generic off-the-shelf optimization package, the paper demonstrated a novel approach to solving certain kinds of real-world problems, contributing a method that have advanced the state of the art in solving a specific optimization problem in the area of economic load dispatch.

Keywords: *economic load dispatch, evolutionary algorithm, optimization, smart mutation*

Cite This Article: Sunny Orike, and Vincent I. E. Anireh, “Smart Evolutionary Algorithm for Static Economic Load Dispatch Optimization in a Thermal Generating Station.” *American Journal of Electrical and Electronic Engineering*, vol. 3, no. 4 (2015): 100-106. doi: 10.12691/ajeee-3-4-3.

1. Introduction

Evolutionary Algorithm (EA) is both a bio-inspired, as well as nature-inspired computational intelligent approach to solving real-life optimization problems [1]. It is a non-deterministic population-based algorithm that is analogous to Darwin’s theory of natural evolution and selection, inspired by nature’s principle of survival of the fittest [2]. In the recent years, EA is an actively growing research area in the entire field of artificial/computational intelligence, especially in solving electrical power problems. Static Economic Load Dispatch (SELD) problem handles a single load optimization period (typically one hour), in which the variables (generator outputs) do not vary with time [3,4,5]. This work presents a novel Evolutionary Algorithm (EA) approach in realizing optimal solutions for SELD problems in the electricity generating station. The method combines a standard EA with a “smart mutation” operator. The key aspect of the method that improves performance is that the operator targets mutation on genes according to their respective contributions to the cost function and penalty violation. The work considers instances of the ELD problem, with practical generator constraints. Violation of either of these constraints introduces the concept of penalties, and these in turn provide the basis for the smart mutation operator. The “Smart” EA was compared with a previously implemented Basic EA (BEA) and with reported results for other recent EAs involving a generator with 6 units.

In traditional EA design, mutation operator is the major source of genetic variation, needed to avoid genetic stagnation [4]. It is a background operator applied to the resulting children solutions after the crossover, and allows new genetic patterns to be introduced, whether desirable or undesirable. Mutation could be viewed as a transition from a current solution to its neighbourhood solution in local search algorithms, as it randomly changes the value of a part of the solution to another. It usually occurs with a low probability, typically between 0 and 1. The process could be very detrimental if the mutation rate is set too high, as it will force the population to adapt to a new environment, and will not necessarily produce optimal individuals due to the instability of the population [2].

In its simplest form of operation, genes undergoing mutation are randomly selected, but they could also be ‘targeted’. Different forms of mutation operators exist for the different encoding methods: Single-gene mutation, *N*-random gene mutation, uniform mutation, boundary mutation and Gaussian mutation, bit-flip mutation, swap mutation, insertion mutation, inversion mutation, scramble mutation, displacement mutation, real-valued mutation, etc [6]. Single gene mutation chooses a gene at random, and changes it to a new value, with a probability, called the “mutation rate”. *N*-random gene mutation repeats the single-gene mutation *N* times. Uniform mutation replaces the value of the selected gene with a uniform random value chosen between user-defined upper and lower limits. Non-uniform mutation increases the probability that the amount of mutation will be close to zero with increased number of generation. Boundary mutation randomly selects a gene, and replaces it with either the lower or upper limit. Gaussian mutation adds a unit Gaussian distributed random value to the selected gene.

Recent researches are progressing towards the area of adaptive mutation, applied to the problem requirements and domain [7]. There are two levels of adaptations in mutation – top level and bottom level [8]. The top-level approach adapts the mutation and crossover ratio in a given EA run [9,10]. In COBRA (COst Based operator Rate Adaptation), developed in [10], the EA swaps a given *k* fixed mutation probabilities periodically between *k* operators, giving highest probability to the operator that has high fitness value. The bottom-level performs a deterministic self-adaptive mutation, probability uniformly or non-uniformly over each gene position [8,11]. The self-adaptive mutation, developed in [11], was achieved by adding a probability vector for each individual. The operator first mutates the mutation probability with itself, and the resulting probability was used to mutate a targeted gene.

The remaining parts of the paper are organized as follows: Section 2 formulates the SELD problem, while Section 3 describes the various processes involved in constraints handling for generating limit, power balance, ramp-rates limit and prohibited operating zones constraints. Section 4 describes the proposed approach in this paper – the smart evolutionary algorithm, including the smart mutation operators and their variants. Section 5 describes the experimental set up, results and discussion of findings, while Section 6 concludes the paper with appropriate recommendations.

2. Problem Formulation

Traditionally, ELD problems are assumed to have quadratic, convex, but smooth cost functions. Consider a thermal generating plant with *N* units, and power outputs Pg_1, Pg_2 to Pg_N, connected through a transmission network, with P_D as the total power demand. Each unit has its own cost function, C_i. The task here is to find the combination of the real power generation for all the units such that the total generation cost, C_T is minimized:

$$Min\ C_T = \sum_{i=1}^{N} C_i = \sum_{i=1}^{N} a_i Pg_i^2 + b_i Pg_i + c_i \qquad (1)$$

[Where: a_i, b_i, c_i are the cost coefficients of unit, i;]

The real power generated by each generator must be between minimum and maximum defined capacities, represented by the following inequality constraints:

$$Pg_{i,\min} \le Pg_i \le Pg_{i,\max} \quad \forall i = 1, 2, ..., N \qquad (2)$$

The total power generated must be balanced, and equal to the sum of total power demand and power loss:

$$\sum_{i=1}^{N} Pg_i = P_D + P_L \quad \forall i = 1, 2, ..., N \qquad (3)$$

The power loss, P_L is calculated using the Kron's formula (also known as the *B*-matrix loss formula) [12], as:

$$P_L = \sum_{i=1}^{N}\sum_{j=1}^{N} Pg_i B_{ij} Pg_j + \sum_{i=1}^{N} B_{0i} Pg_i + B_{00} \qquad (4)$$
$$\forall i,\ j = 1, 2, ..., N$$

[Where: B_{ij} is the ij^{th} element of the loss coefficient square matrix of the same dimension as Pg_i; B_{0i} is the i^{th} element of the loss coefficient vector of the same length as Pg_i; B_{00} is the loss coefficient constant].

3. Constraints Handling

A functional generator has four operational constraints: power balance, generating limit, ramp-rates limit and prohibited operating zone constraints. Power balance constraint is global (applying to all the generating units), while others are local to each unit. The smart mutation operator focuses mutation on units contributing mostly to cost and violates either of the local constraints. One of the ways to investigate the overall performance of an algorithm is by evaluating its constraints handling capabilities. The random initialisation of chromosomes (potential solutions to a problem) sets the genes (outputs of generating units) of each chromosome in the initial population to random double numbers representing the generators' outputs, between their minimum and maximum generating limits. During fitness evaluation (computation of the cost function) of each chromosome, various checks are performed to ensure that the units' outputs obey all operational constraints. Violation of any of the constraints constitutes a penalty, which augments the initial objective cost function to form the generalised fitness function:

$$Min\ C_T = \begin{cases} C_T \\ C_T + total\_penalty \end{cases} \qquad (5)$$

This helps guide the search process towards repairing that solution. A description of how the four operational constraints are handled is presented here:

i). *Generating Limit:* A random small positive number less than 1 (scaling factor), ensures that outputs of the generating units are within the allowable minimum and maximum limits:

$$Pg_i = Pg_i^{\min} + Math.random() * (Pg_i^{\max} - Pg_i^{\min}) \qquad (6)$$

ii) *Power Balance:* If the equality constraint of (3) is not satisfied, a penalty factor, q_1, is used to normalize and maintain an overall power balance, ensuring that the terms in the bracket equal to zero:

$$Penalty\_pb = q_1 \left(\sum_{i=1}^{N} Pg_i - P_D - P_L \right)^2 \qquad (7)$$

iii) *Ramp-rates Limit:* The generating units' ramp-rates limit restricts their operating range in a given generation period. A ramp-rate is the rate of change in output from a power plant. It is the amount of load added to the generating plant to keep it from being "overloaded". Based on these rates, the subsequent outputs of the units should be within their ranges, with (2) modified as:

$$\max(P_{i\min}, P_i^0 - DR_i) \le P_i \le \min(P_{i\max}, P_i^0 + UR_i) \qquad (8)$$

The power output of a unit at the current time depends on the output at the previous time, ramp up value (*UR*) and ramp down value (*DR*) of the generator. The power output of a unit due to the ramp-rates limit is defined as follows:

$$Pg_{rr\lim} = \begin{cases} Pg_{i,t-1} - DR_i, & Pg_{i,t} < Pg_{i,t-1} - DR_i, \\ Pg_{i,t-1} + UR_i, & Pg_{i,t} > Pg_{i,t-1} + UR_i, \\ Pg_{i,t}, & otherwise \end{cases} \quad (9)$$

If the inequality constraint of (8) is not satisfied, a penalty factor, q_2, is assigned to the affected units outside the feasible regions, using:

$$Penalty\_Pg_{rr\lim} = q_2 \left(\sum_{i=1}^{N} Pg_i - Pg_{rr\lim} \right) \quad (10)$$

iv) Prohibited Operating Zones: Practical operations of the power plant involve adjusting the power output of the units such that they must be outside the prohibited zones. The prohibited operating zone penalty function counts the number of units that fall within such prohibited zones, according to the following rules:

$$PZ_{i,k} = \begin{cases} 1, & if\ Pg_i\ violates\ prohibited\ zone \\ 0, & otherwise \end{cases} \quad (11)$$

If the prohibited zones are not violated, the ELD is solved in a straightforward process, otherwise, a number ‘1’ is assigned to such occurrence, and the constraint is processed using:

$$Penalty\_pz = q_3 \sum_{i=1}^{N} PZ_{i,k} \quad (12)$$

[Where: q_3 = penalty factor, k = number of prohibited zones in i, N = number of generating units].

4. Smart Evolutionary Algorithm

The approach of capturing gene-specific contributions to costs of generating electrical power, and using this information to help target the mutation operator, is what is referred to in this paper as “smart evolutionary algorithm” (SEA). The method basically combines a standard EA with a smart mutation operator. In the standard EA, single mutation per chromosome is used. Occurring at a low probability called mutation rate, it chooses a gene at random and changes it to a new value. It works as a background operator, introducing new genetic patterns in the population by randomly modifying some building blocks to maintain population diversity, thereby helping solutions to escape local optima. This mutation scheme has severally been used and shown in the literature to produce near-optimal results [13], including a real-life virology application [14]. The generic SEA pseudo-code for SELD is shown below:

```
START
DEFINE parameters and INPUT data
INITIALIZE a random population of chromosomes
  FOR i = 1 to N (N = No of Generating Units)
    $Pg_i = Pg_{i\_\min} + Math.Random * (Pg_{i\_\max} - Pg_{i\_\min})$
  END FOR
EVALUATE Objective function (For each chromosome in the population)
FOR i = 1 to N
        $f(Pg_i) = a_i Pg_i^2 + b_i Pg_i + c_i$
    Check for constraints violations
    Violation of a constraint leads to penalty treatment, defined as:
        $Min\ F_c = \begin{cases} f(Pg_i) \\ f(Pg_i) + penalty(Pg_i) \end{cases}$
    IF no constraint is violated,
        Penalty = 0
        Fitness function = Objective function
    ELSE
        Fitness function = Objective function + Penalty
END IF
END FOR
WHILE (stopping criteria is not reached)
    From the entire population
    SELECT parent pairs for breeding
    CROSSOVER the genes of selected parents
    Create an array to return children
    Crossover parents to create children
    Return the children in an array
   Perform SMART MUTATION
  EVALUATE
  REPLACE parent population with children population
  Perform ELITISM (Keep a percentage of best individuals)
 FOR i = 1 to Population size
Sort Chromosomes according to their fitness (highest to lowest)
    Calculate Number of Elites (based on elitism rate)
    Copy Elites onto next Generation
  END FOR
END WHILE
OUTPUT (The optimum/best compromising solution vector)
 FOR i = 1 to N
   $Pg = [Pg_1, Pg_2, Pg_3, \ldots Pg_N]$
 END FOR
 Output Total Generation = Sum of all Generators' Output
 Output Total cost = Sum of Costs of generation
STOP.
```

4.1. The Smart Mutation

The total number of units selected for mutation in addition to the highest cost producing one depends on the number of local constraints violated. Therefore a minimum of one and maximum of three genes are involved. Perhaps in some problem cases, either or all of the local constraints may not be present. In some other cases, the unit with the highest cost may also violate a constraint, while in some other cases, no constraint may be violated. But where violation exists, the numerical (penalty) value augments the problem’s objective function to form a generalised fitness function.

The magnitude of the constraints as well as the cost produced by each of the units contributes to the overall fitness. As the penalties values gradually reduce to 0, total cost equals total fitness, which is the optimal value. By targeting the unit with the highest cost, the mutation operator attempts to minimise the total cost of producing an optimal power. It was found out that reducing the values of those units reduces over generation of power, and consequently minimises power loss. Mutating the units that violate local constraints has a tendency of forcing them into the feasible regions. Besides, the small random deviation subtracted from the units ensures that the changes introduced are not too significant, thereby distorting the generating unit. The following is a pseudo code of the smart mutator.

```
START
INITIALIZE costs
Calculate the costs produced by all the units
SET cost of the first unit (i = 1) as the highest cost
FOR i = 2 to N (where N = number of generating units)
        IF (cost[i] > highest cost)
                Highest cost = cost[i]
                Highest position = i
        END IF
        $Pg_i$ = value of unit at highest position
        Mutate this $Pg_i$ by subtracting a small random deviation from the unit:
        $Pg_i (new) = Pg_i – (Math.random () * Pg_i)$
        (Where: Math.random () is a random number between 0 and 1, e.g. 0.2)
        Replace $Pg_i$ with $Pg_i$ (new)
        Check for violation of local constraints
        Ramp-rate limits violation:
        IF Generation decreases
                $Pg_i = (Pg_{i-1} – DR)$
                IF Generation increases
                        $Pg_i = (Pg_{i-1} + UR)$
                        (Where: $Pg_i$ = current value; $Pg_{i-1}$ = previous value; UR = up-ramp rate limit)
        ELSE
                $Pg_i = Pg_{i-1}$
        END IF
    END IF
  IF ($Pg_i$ < Max (lower limit, $(Pg_{i-1} – DR)$)) OR ($Pg_i$ > Min (upper limit, $(Pg_{i-1} + DR)$))
  (Where: lower limit and upper limit = legal lower and upper limits of the unit)
                $Pg_i$ violates ramp-rates limit
  Mutate this $Pg_i$ by subtracting a small random deviation from the unit:
        $Pg_i (new) = Pg_i – (Math.random () * Pg_i)$
        END IF
        Replace $Pg_i$ with $Pg_i$ (new)
        Prohibited operating zones violation
  FOR k = 1 to $z_i$ (where: $z_i$ = number of prohibited zones of unit, i)
        IF ($Pg_i > Pg^l_{i,k}$) AND ($Pg_i < Pg^u_{i,k}$)
        (Where: $Pg^l_{i,k}$ and $Pg^u_{i,k}$ are the lower and upper bounds of the kth prohibited zone)
  Mutate this $Pg_i$ by subtracting a small random deviation from the unit
            $Pg_i (new) = Pg_i – (Math.random () * Pg_i)$
  Assign the number “1” for every $Pg_i$ that violates prohibited operating zone
    (i.e. counting every occurrence)
                END IF
          END FOR
          Replace $Pg_i$ with $Pg_i$ (new)
  Ensure that all the units’ outputs are within feasible limits (Generating limit constraint):
  $Pg_i = Pg_{i\_min} + (scaling\_factor * (Pg_{i\_max} – Pg_{i\_min}))$
END FOR
STOP
```

4.2. SEA Variants

Several variants of the smart mutator are possible. In this paper, an investigation is made of the following three smart evolutionary algorithms: SEA1, SEA2 and SEA3, resulting from three distinct variants of the smart mutator. SEA1 involves the use of tournament selection based on the penalty values (costs and numerical values of the constraints) to decide which gene to mutate. It works in direct analogy to the tournament method of selecting parents from an initial population to go into breeding in order to generate an offspring during an evolutionary process [4]. But while the EA selection method chooses chromosomes (the entire generating units) from the population uniformly at random, with replacement, and the fittest of those individuals is the one returned as selected, the SEA1 chooses genes from the chromosome, by targeting those with the highest operating cost and violate constraint(s). This is the basis of the new chromosome, a potential solution to the problem. The number of units selected (tournament size) as well as the size of the problem case has combined effect on selective pressure [4,5]. In SEA2, there is a mutation probability, called smart mutation probability, different from the normal mutation rate of an EA. The smart mutation probability biases the mutation operation, such that when the probability is met, smart mutation is done; otherwise, a single-gene, uniform random mutation is done. This gene-specific mutation probability in SEA2 is also a random number between ‘0’ and ‘1’. In contrast with the mutation rate which usually occurs with a much lower probability, and aims to maintain diversity in the entire population of individuals; the smart mutation probability is a fixed, higher-level probability operator that targets a particular gene based on the criteria of merit; uniformly or non-uniformly, deterministically and adaptive [8], [11]. SEA3 is an extension of SEA2, but rather than having a fixed value of the smart mutation probability, it is computed (as the ratio of generation number over the maximum generation). Therefore, the mutation probability value starts at ‘0’, and gradually moves to ‘1.0’ in a linear fashion towards the maximum number of generations. This is a simple linear adaptation that works by linearly increasing the smart mutation probability from beginning to end of an optimization run.

5. Experiments, Results and Discussion

The work described in this paper is concerned with optimization of electrical power generator outputs. Optimization is a way of making things better; a method of adjusting the inputs to, or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output/result; an art of allocating resources to the best possible effect [15]. Optimization is analogous to the root-finding process in calculus, but while the latter searches for zeros of a function, the former finds zeros of the function derivatives. But a major difficulty with optimization, unlike root-finding is determining if a given minimum or maximum is a global optimum (the best possible solution available) or a local optimum. Three SEA variants were investigated and compared with an earlier developed BEA [4] on the basis of efficiency, validity and robustness of the algorithms in a problem case involving 6 generating units. In [3], the SELD problem was solved using Differential Evolution (DE), allocating power output to 6 thermal generators, taking into account the effects of transmission losses. Total load demand was set at 283.40MW; the detailed parameters and loss coefficients can be retrieved from [3][4]. Based on available data in this problem case, the main constraints which constitute the source of penalties are generating limits and power balance constraints. Augmenting the penalty function to the original objective cost function of (1) yields the following generalised fitness function:

$$C_T = \sum_{i=1}^{N} C_i + q_1 \sum_{i=1}^{N} (Pg_i - P_D - P_L)^2 \qquad (13)$$

Where: q_1 is a penalty factor which normalizes the power balance, assigning a high cost of penalty to affected ones far from the feasible region [3]. Detailed evolutionary tuning was carried out in an earlier work to select optimal values for the genetic parameters (crossover rate, population size, tournament size and mutation rate) in [5]. As in SEA1, SEA2 operates a single-gene mutation per solution of rate 0.01 in addition to the smart mutation probability of 0.6. Table 1 gives the tuned values of the experimental parameters, involving SEA1, SEA2 and SEA3, with results shown in Table 2.

Table 1. Experimental Parameters and Values [5]

Parameters	Values
Population size	100
Tournament size	2
Crossover rate	0.7
Mutation rate	0.01
Mutation probability (in SEA2)	0.6
No of Generations	100
Elitism Rate	10%
No. or runs	30

Table 2. Summary of Results, Averaged over 30 Runs in each of SEA1, SEA2 and SEA3 Approaches

	SEA1	SEA2	SEA3
Av Cost ($/h)	738.7	749.5	744.5
Std Dev	13.7	13.3	10.6
Min Cost ($/h)	709.6	711.9	710.1
Max Cost ($/h)	759.2	767.0	759.0

The results of SEA1, SEA2 and SEA3 were compared with a previously developed BEA [4]; and two reported similar EA approaches - Differential Evolution (DE) and Genetic Algorithm (GA) [3], tested on the same problem case. Table 3 summarizes the comparison results based on the resources allocation to the units from the best of 30 independent runs of algorithms BEA, SEA1, SEA2 and SEA3 (the number of runs for DE and GA were not reported). This shows superior performance of SEA1, SEA2 and SEA3 in terms of both lower generation costs and lower power losses.

Table 3. Resources Allocation in the Best of 30 Runs of SEA1, SEA2, SEA3, BEA, DE and GA

Units	SEA1	SEA2	SEA3	BEA[4]	DE[3]	GA[3]
1	151.3	169.0	130.4	171.6	177.5	179.4
2	24.7	30.5	41.1	49.3	48.6	44.2
3	48.9	40.8	49.6	22.6	20.9	24.6
4	23.5	11.9	27.6	21.2	21.6	19.9
5	15.7	15.8	19.5	12.7	12.5	10.7
6	19.9	15.7	16.2	14.2	12.0	14.1
Power Gen(MW)	284.0	283.7	284.4	292.1	293.2	292.9
Power Dem(MW)	283.4	283.4	283.4	283.4	283.4	283.4
Loss(MW)	0.6	0.03	1.0	8.7	9.8	9.5
Cost ($/h)	709.6	711.9	710.1	801.3	803.1	803.7

A comparison was made of the penalties handling capabilities of BEA, SEA1, SEA2 and SEA3 as shown in the costs and penalties convergence characteristics of Figure 1 to Figure 4. The Fitness axes are the values of total costs and penalties. Starting with randomly generated populations at generation 0, the values of the costs and penalties gradually converge smoothly to the respective optima as generation increases in each of the four algorithms.

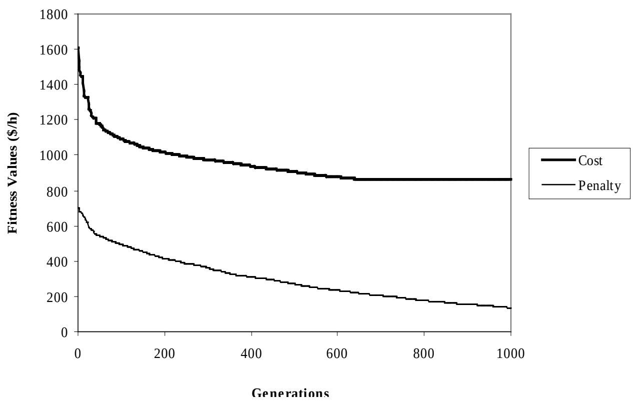

Figure 1. Cost and penalty convergence characteristics of BEA

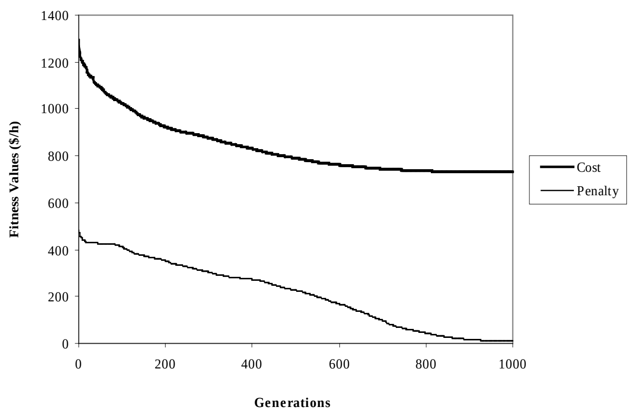

Figure 2. Cost and penalty convergence characteristics of SEA1

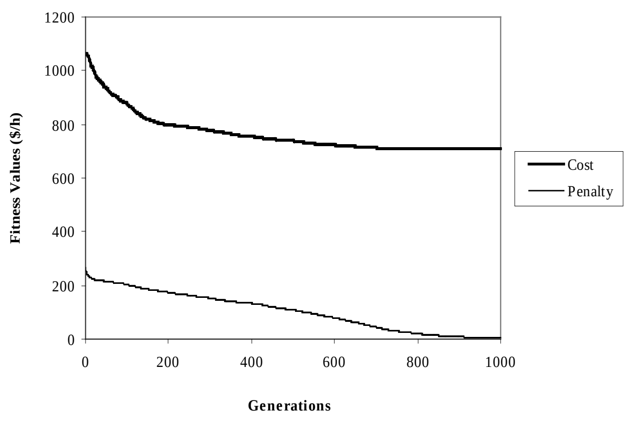

Figure 3. Cost and penalty convergence characteristics of SEA2

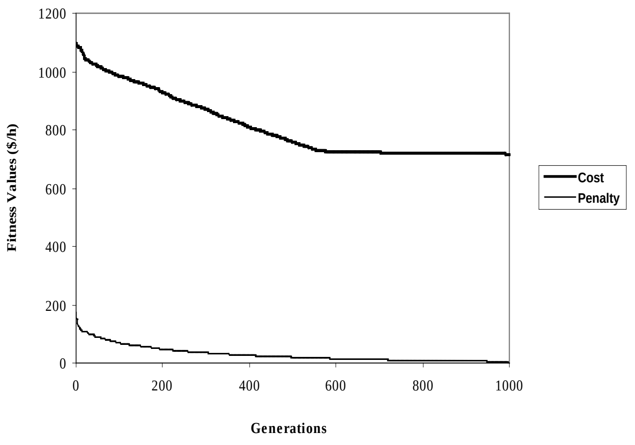

Figure 4. Cost and penalty convergence characteristics of SEA3

There is a great similarity in the trend of cost curves of BEA and SEA1, but the major difference is in the penalty curves. From the results, BEA is not very efficient in reducing penalty violations, unlike SEA1, SEA2 and SEA3. As the value of the total penalties gradually reduce to 0.0, total cost equals total fitness, which is the optimal value. This contributes to the 'smartness' of the three SEAs. Moreover, the best results from BEA have a considerably higher total cost than SEA1, SEA2 and SEA3. From the results, SEA3 shows the most active performance in reducing penalty violations. This evidence suggests that SEA3 will be capable of optimizing larger scale problems. However, the generation cost of $709.6/h from the best resources allocation of SEA1 (Table 3) is the lowest value seen in the literature to date for this problem, while meeting load demand.

The simulation results were compared with those reported for some recent alternative EAs, where the SEAs exhibited superior performances. On the basis of the optimal/best resources allocation, they were all better in terms of both lower generation costs and lower power losses. SEA1 achieved a lower average generation cost of $738.7 (Table 2) over SEA2 and SEA3, and also the minimum generation cost of $709.6/h from the best resources allocation (Table 3) is the lowest value seen in the literature to date for this problem. However, a minimized power loss of 0.03MW was realized using SEA2, meaning that it has the greatest potential of meeting power demand; and reducing both over/under generation of electricity. But on the basis of constraints handling capability, SEA3 proved the best optimization approach from the cost and penalty convergence characteristics of Figure 4.

6. Conclusion

This paper proposed a Smart Evolutionary Algorithm (SEA), which combines a standard EA with a smart mutation approach for the Static Economic Load Dispatch (SELD) optimization problem in a thermal generating station. The operator focused mutation on genes contributes mostly to costs and penalty violations, while obeying operational constraints. Three variants of smart mutation operator were developed, leading to SEA1, SEA2 and SEA3, on a benchmark case involving a generating station with 6 units. This novel approaches to SELD were shown to outperform all previously published EA approaches, on the basis of the common published test problem used in the literature in terms of reduced cost of generation, minimized power loss, optimal resources allocation constraints handling capability. The SEA3 showed the most active performance in reducing penalty violations, an evidence that it will perform well in large scale test problems. Standard T-test (one tailed) with significance level $p < 0.1$ (confidence level 90%) shows no significant difference statistically between the three approaches for this problem. However, the power dispatch considered in this paper is for thermal plants only (driven by heat released from burning of fossil fuels - coal, petroleum, natural gas). One potential area of future work is to extend this approach to handle generating stations that use multiple fuel types.

References

[1] Orike, S., "Computational Intelligence in Electrical Power Systems: A Survey of Emerging Approaches," *British Journal of Science*, 12 (2). 23-45. April. 2015.

[2] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[3] Sayah, S. and Zehar, K., "Using Evolutionary Computation to Solve the Economic Load Dispatch Problem," *Leonardo Journal of Sciences*, 12. 67-78, Jan-Jun. 2008.

[4] Orike, S. and Corne, D.W., "Improved Evolutionary Algorithms for Economic Load Dispatch Optimisation Problems," in *12th UK Workshop on Computational Intelligence (UKCI)*, Edinburgh, 5-7 Sept. 2012, IEEE.

[5] Orike, S., "Investigating the Effects of Evolutionary Parametric Tuning for Static Economic Load Dispatch Problems," *Asian Engineering Review*, 1(2). 26-35, Nov. 2014.

[6] Hasan, B.H.F., and Saleh, M.S.M., "Evaluating the Effectiveness of Mutation Operators on the Behaviour of Genetic Algorithms Applied to Non-deterministic Polynomial Problems," *Informatica*, 35. 513-518, 2011.

[7] Korejo, I., Yang, S., and Li, C., "A Comparative Study of Adaptive Mutation Operators for Genetic Algorithms," in *8th Metaheuristics International Conference*, Hamburg, Germany, July 13-16, 2009.

[8] Yang, S., "Statistics-Based Adaptive Non-Uniform Mutation for Genetic Algorithms," in *2003 Genetic and Evolutionary Computation Conference, July 9- 11, 2003, Chicago, USA.*

[9] Julstrom, B., "What have you done for me lately? Adaptive Operator Probabilities in a Steady-State Genetic Algorithm," in *6th Conference on Genetic Algorithms*, San Mateo, CA, USA, 1995, Morgan Kaufmann, 81-87.

[10] Corne, D., Ross, P., and Fang, H.L., "Genetic Algorithm Research Note 7: Fast Practical Evolutionary Timetabling," *Technical Report*, Department of Artificial Intelligence, University of Edinburgh, UK, 1994.

[11] Bäck, T., “Mutation Parameters,” in *Bäck, T., Fogel, D.B., and Michalewicz, Z. (eds.), Handbook of Evolutionary Computation*, E1.2.1- E1.2.7, Oxford University Press, 1997.

[12] Wood, A.J., and Wollenberg, B.F., *Power Generation, Operation and Control*. 2nd Ed., John Wiley, 2012.

[13] Woodward, J., and Swan, J., “The Automatic Generation of Mutation Operators for Genetic Algorithms,” in *2012 Genetic and Evolutionary Computation Conference*, Philadelphia, USA, July 7-11, 2012.

[14] Tao, W., Xu, C., Ding, Q., Li, R., Xiang, Y., Chung J., and Zhong, J., “A Single Point Mutation in E2 Engances Hepatitis C Virus Infectivity and Alters Lipoprotein Association of Viral Particles,” *Journal of Virology*, 395. 67-76, Dec. 2009.

[15] Haupt, R.L., and Haupt, S.E., *Practical Genetic Algorithms*. John Wiley, 2004.