Solving Optimal Load Dispatch Problem Using Evolutionary Intelligence with Degree of Violation Approach

Abhishek Reddy Sheri
Electrical and Electronics Engineering Department
Mahatma Gandhi Institute of Technology
Hyderabad, India
palavarapuprakash44@gmail.com

Ramchandra Reddy Annadi
Electrical and Electronics Engineering Department
Mahatma Gandhi Institute of Technology
Hyderabad, India
arcreddy13@gmail.com

***Abstract*—The aim of this study is to reduce the cost of power generation and minimize constraint violations using three evolutionary algorithms. To achieve this, we designed a fitness function that assigns higher fitness values to solutions with low cost and low constraint violation. The fitness function incorporates cost and violation terms in its denominator, enabling the algorithms to converge towards the best solution. Simulations were conducted on three test systems to determine the algorithm that strikes a suitable balance between cost and violation reduction for solving the optimal load dispatch problem. Upon comparing the simulation results, we found that the artificial bee colony algorithm effectively minimized both the cost and violation metrics in all three test cases.**

Keywords — Optimal load dispatch, Constraint Violation, evolutionary algorithms, Artificial bee colony algorithm, Binary coded genetic algorithm

I. INTRODUCTION

The main goal of a Power System (PS) network with numerous equipment of large rating is to supply required power to load continuously, if done in an economically optimal way is termed as Optimal Load Dispatch (OLD). Generally, OLD can be formulated as an optimization problem and can be solved using evolutionary algorithms [1]. Evolutionary algoriths can also be used for evaluationg optimal performance with aid of PSCAD to determine the use of best devices or best locations in the power system especially in the UHV AC systems [2-4]. There are many methods to solve OLD problem, some of the traditionally used methods like lambda iteration method, gradient method and base point and participation method assumes that cost curves of generating units are ascending piecewise-linear functions, regrettably, this assumption might make these methods impractical due to their non-linear characteristics when applied to real-world systems [5], also these approaches have a difficulty in finding global optimal solution and stuck in local optimal solution [6]. Moreover, other approaches like Dynamic Programming (DP) will have large dimensions which increases computational complexity [5], the alternate approach to find solution for OLD is using evolutionary computation techniques like Particle Swarm Optimization (PSO) [5], Artificial Bee Colony (ABC) algorithm [7], Binary coded Genetic Algorithm (BGA) [5], Grasshopper

Algorithm (GOA) [3], Biogeography-Based optimization (BBO) [6], and many more have successfully solved OLD problem and obtained good convergence towards optimal generating cost. To solve the OLD problem, it is necessary to satisfy the power balance equation, which is the equality constraint. The focus of our investigation is to solve OLD problems in a new approach called *Degree of Violation approach* for two Evolutionary Algorithms (EAs) namely - ABC and BGA. The degree of violation approach for EA is a new way to solve OLD, which drives the algorithm towards the solution, which is most feasible, by calculating the constraint violations of solutions (population) in every iteration, basing on which the fitness value is assigned for each solution, finally the best fit solution after all iterations is returned as optimal solution. Section II of this paper describes the problem statement and constraints briefly for OLD problem. The constraint violation in OLD problem is discussed in section III along with necessity of minimizing the violation and degree of violation to which it should be minimized. The fitness function that helps to minimize the constraint is discussed in section IV, both III and IV sections are heart and soul of the paper. Three EAs are briefly discussed in section V. In section VI, use of python for optimizing OLD is discussed briefly. The results obtained are discussed in section VII. The conclusion drawn from the paper is discussed in section VIII.

II. PROBLEM DESCRIPTION

A. Optimal Power Dispatch Problem

To schedule the power generations of individual units with lowest possible fuel cost using optimization techniques, one should define problem parameters. The problem parameters include, cost curve coefficients of individual unit, power limits of every generator, load demand and B – coefficients of the system. The objective function is defined as follows:

$$C_i = a_i + b_i P_i + c_i {P_i}^2 \tag{1}$$

The objective function in eq. (1), is a cost curve of i^{th} generating unit which is a quadratic polynomial. Where, a, b and c are cost curve coefficients of i^{th} unit.

B. Inequality Constraints

In a power station of N generating units, every generator has power limits, where generation should be done within those limits.

$${P_i}^{min} \leq P_i \leq {P_i}^{max} \tag{2}$$

C. Equality constraints

The power station should meet the load and losses in transmission lines, this is mathematically given by

$$\sum_{i=0}^{N} P_i = P_D + P_L \tag{3}$$

where, P_D is the load demand, P_L is the loss expressed in terms of power generations and B - coefficients.

$$P_L = \sum_{i=0}^{N} \sum_{j=0}^{N} P_i P_j B_{ij} \tag{4}$$

III. CONSTRAINT VIOLATION

Constraints are conditions or limitations that must be satisfied in order to find a valid solution. When a solution violates one or more of these constraints, it is considered to have a constraint violation. *Constraint Violation* (CV) refers to the amount of deviation or non-compliance with the constraints defined in the problem statement or model. The violation in optimal load dispatch is given by eq. (5).

$$CV = (P_D + P_L) - \sum_{i=0}^{N} P_i \tag{5}$$

A. Necessity of mimimizing violation

The original problem at hand revolves around a power balance constraint, which is mathematically represented by equation (3). Equation (5) quantifies the extent of the constraint violation. Let's consider a violation value of 1, indicating a shortage of 1 MW of power. This particular deficit, regardless of the specific value, should not be taken lightly, as it signifies a significant shortfall when considering the units involved. In this scenario, the system is actively supplying the existing load, but it falls short of meeting the overall demand by the specified amount.

In this context, the shortage of power, whether it is 1 MW or any other value, corresponds to the power needed to sustain a significant number of households. Assuming an average daily household usage, neglecting the constraint violation would result in numerous households being deprived of the necessary power supply to meet their needs. Nowadays, electricity has become a fundamental necessity, not only for individual households but also for the smooth functioning of various small, medium and large-scale industries and businesses. Overlooking the constraint violation in this scenario could lead to substantial losses for the economy and hinder overall development. Being deprived of adequate power supply, regardless of the specific shortage, can have far-reaching consequences.

In conclusion, regardless of the specific shortage, any deficit in power supply due to a constraint violation should not be underestimated. Neglecting such an issue would result in many households being deprived of the necessary power supply, hindering their daily lives and well-being. Additionally, it would have adverse effects on industries, businesses, education, and healthcare sectors, leading to economic losses and impeded progress. Addressing constraint violations and ensuring equitable and reliable access to electricity becomes paramount to support the overall welfare and advancement of society.

B. Degree of Violation to be Considered

In the case of the OLD problem, it is crucial to minimize the constraint violation. However, it is equally important to consider the extent to which it gets minimized. This is because the violation is measured in the same units as power, namely megawatts (MW), which carries significant weight regardless of its specific value. Therefore, it becomes necessary to reduce the violation to a degree of 10^{-4} to 10^{-6} which translates MW into tens of watts or watts.

Algorithms that possess the capability of reducing the violation by an order of 10^{-7} are not necessarily required. Instead, the objective should be to design the algorithm in such a way that it effectively minimizes the violation to around 10^{-4}. This approach helps strike a balance between minimizing the degree of violation and avoiding excessive computational time for the

algorithm. By adopting this approach, it can be ensured that the constraint violation is significantly reduced while maintaining a reasonable computational efficiency.

IV. VIOLATION BASED FITNESS EVALUATION

A. Adpoted Fitness Function

$$Fit(F,V) = \frac{1}{(1+F)\times(1+V)^n} \tag{6}$$

As the goal is to minimize the cost as well as violation, both cost and violation are in denominator of eq. (6). Here, F is objective function or cost value of a particular solution and V is the violation of equality constraint which, adding positive power (i.e., n in (6)) to the violation makes fitness function more sensitive towards violation. This power can be varied based on different system conditions; one can adjust the value of n by analysing the magnitude of violations obtained. By observing eq. (6), if cost F increases, then total fitness value decreases and same for violation V, where n is the importance factor given for constraint violation.

So, the solution with less cost and violation will have more fitness. If the product of the denominator is negative, then eq. (7) or eq. (8) can be used as fitness function.

$$Fit(F,V) = |F \times V| \tag{7}$$

$$Fit(F,V) = |F + V| \tag{8}$$

B. Sensitivity of Fitness Towards Violation

The fitness function defined by eq. (6), is modelled in a way that it's sensitivity towards violation can be adjusted, also if sensitivity of fitness function towards change in violation is more, the oscillations while converging might increase and lead to late convergence, so, eq. (6) with higher value of n is not preferred for every algorithm. To understand the sensitivity of fitness towards violation, it is necessary to observe the variation of fitness with respect to change on violation by keeping the cost constant. Assume the value of cost F is constant and equal to 10 and vary the violation from 0.00005 to 0.0001, consider 50 values in that range, let us observe the various conditions i.e., when $n = 1, n = 4, n = 7$ and $n = 10$ from the plot given in fig.1.

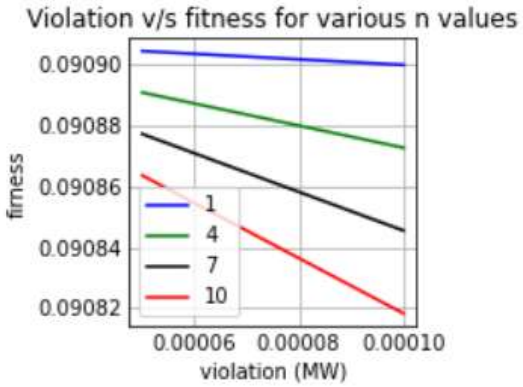

Fig. 1. Variation of Fitness Value with respect to Violation for different n values

From fig.1 it can observed that, for smaller n values, the slope is less and on increasing the value of n, slope of the curves increased, which indicates that, higher the value of n makes fitness function more sensitive towards violation. That means, for higher value of n, a small change in violation changes the fitness function largely.

The highly sensitive fitness function might cause oscillations, also greater the n value, more will be the degree of violation. That means, highly sensitive fitness function might minimize the violation to a degree lower than 10^{-6}, (10^{-7} , 10^{-8} and lower degrees) which is not computationally efficient as mentioned in part B of section III. Hence it is important to choose the value of n such that it should balance the degree of violation and sensitivity of fitness function

V. EVOLUTIONARY ALGORITHMS

A. Artificial Bee Colony Algorithm

The Artificial Bee Colony algorithm is a nature-inspired optimization technique that draws inspiration from the foraging behaviour of honey bees, introduced by Dervis Karaboga in 2005 [7]. The ABC algorithm employs the concepts of employed bees, onlooker bees, and scout bees to explore the search space and find the best solutions. Employed bees explore the vicinity of known solutions and exploit them to improve their current positions [8]. Onlooker bees select promising solutions based on their fitness values and share information with employed bees [9]. Scout bees periodically explore new areas of the search space to discover potentially better solutions and maintain diversity within the population [13]. At the start of the ABC algorithm, an initial population of randomly generated solutions, called food sources, is created within the problem's search space. Each food source represents a potential solution to the optimization problem. Employed bees evaluate the fitness of these food sources using the objective function of the problem and iteratively update their solutions by exploring neighbouring solutions [8]. Onlooker bees choose food sources based on their fitness values and communicate these selections to employed bees, who then adjust their positions accordingly [9]. This process facilitates information exchange and exploitation of promising regions in the search space. The ABC algorithm continues to iterate, updating and refining the population of solutions. Termination occurs when a predefined stopping criterion is met, such as reaching a maximum number of iterations or achieving a satisfactory solution.

In this paper, ABC algorithm with little modifications is proposed, in which convergence is observed even without scout bee phase. The algorithm for adopted ABC is shown below

1. Initialize random solutions within search space
2. Calculate objective function and constraint violation using (1) and (5) respectively.
3. Substitute these values in (6) to evaluate fitness function.
4. Start employed bee phase, for each bee
    a. Modify the solution using (13)

$$X_{old} = X_{new} + \Phi\left(X_{partner} - X_{old}\right) \tag{13}$$

    b. Evaluate the new fitness
    c. Compare old and new fitness
    d. Discard the worst solution and continue with good solution.
5. Start onlooker bee phase, for each bee,

a. Calculate the probability using equation

$$Prob_i = 0.9 + \frac{Fit_i}{Fit_{max}} + 0.1 \quad (14)$$

b. Generate random value r within 0 and 1.
c. Modify solution if r greater than probability.

6. Check if termination condition is satisfied,
7. Go to step 4 if termination not met, else stop execution.

B. Binary Coded Genetic Algorithm

The Binary Coded Genetic Algorithm is a popular optimization technique that utilizes the principles of genetic algorithms to solve problems with binary-encoded representations. It is widely used for tackling a range of optimization problems, including those in engineering, computer science, and other domains. The BGA effectively combines the power of evolutionary search and genetic operators to efficiently explore the search space and find optimal solutions. In the BGA, the search space is represented as a binary string, where each bit represents a decision variable or a feature of the problem [14]. The binary representation allows for efficient manipulation and evaluation of potential solutions. The algorithm begins by initializing a population of binary strings randomly. The fitness of each individual in the population is then evaluated based on an objective function specific to the problem at hand. The core idea of the BGA lies in the iterative process of evolution, which consists of selection, crossover, and mutation operations. During the selection phase, individuals with higher fitness values are more likely to be selected for reproduction [15]. Crossover involves combining genetic information from two parent individuals to create new offspring. This operation mimics the biological process of recombination and promotes exploration of the search space. Mutation introduces random changes in the offspring's genetic information to enhance exploration and prevent premature convergence.

The BGA continues to iterate through these steps, allowing the population to evolve over generations. Through the process of selection, crossover, and mutation, the algorithm effectively explores the search space [5]. The termination criterion can be based on a maximum number of iterations or reaching a satisfactory solution. The algorithm for BGA is given as below:

1. Initialize random solutions (binary strings).
2. Evaluate the decimal equivalent of binary strings.
3. Calculate the actual values corresponding to decimal equivalents within search space.
4. Calculate objective function and constraint violation using (1) and (5) respectively.
5. Substitute these values in (6) to evaluate fitness function.
6. Complete the selection process using roulette wheel selection operator.
7. Generate off springs using single point crossover.
8. Complete bit-wise mutation to generate mutated off springs.
9. Join the initialized solutions and mutated offspring solutions to pick the best fit solutions.
10. This new list of solutions will be sent to step 6.
11. Repeat the process until termination criterion was met.

VI. RESULTS

The ABC and BGA algorithms are implemented in Python to address three distinct test systems that have been extensively utilized in prior research studies. These test systems are specifically designed to encompass varying numbers of generating units, enabling the evaluation of algorithm performance across a diverse range of scenarios. By employing these different test systems, it can be effectively assessed how well each algorithm performs under varying number of generating units. The fitness function mentioned in eq. (6) is considered for all algorithms with magnitude of n equal to unity $(n = 1)$.

A. Case 1 - Three Units System

The data for case 1 was taken from [14], where among three units, one is coal-fired and two others are oil-fired units. The load demand that is aimed to meet was 850MW. Simplified loss equation is (9), which is given below,

$$P_{loss} = 0.00003P_1^{\,2} + 0.00009P_2^{\,2} + 0.00012P_3^{\,2} \tag{17}$$

For this case, (6) is considered as fitness function with n equal to one for ABC and BGA.

TABLE I: 3 Unit System Data

Generating unit	a	b	c	P_{min} (MW)	P_{max} (MW)
1	561	7.92	0.001562	600	150
2	310	7.85	0.001940	400	100
3	78	7.97	0.00482	200	50

The graph plotted for cost with respect to iterations is shown in Fig. 2 and violation with respect to iterations is shown in Fig. 3, both Fig. 2 and Fig. 3 compare the convergence of three algorithms in cost and violation aspects respectively.

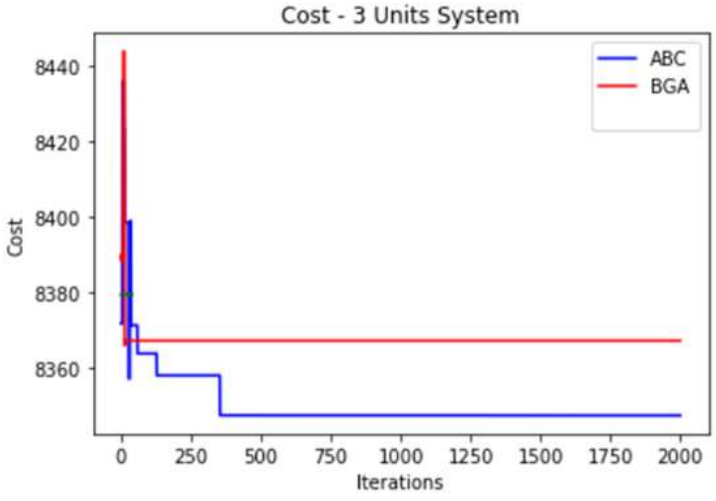

Fig. 2. Convergence of cost for 2 algorithms (Case-1)

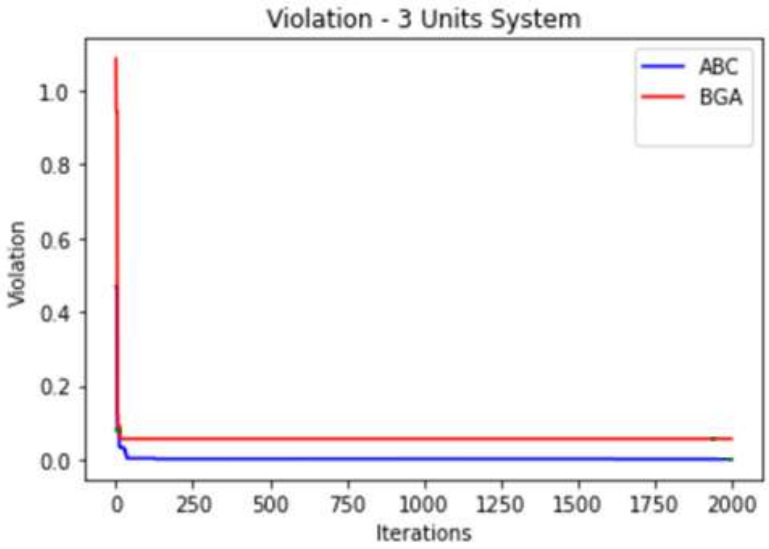

Fig. 3. Convergence of violation for 2 algorithms (Case-1)

From Fig. 2 and Fig. 3, it can be concluded that ABC shows better results in minimizing the cost and violation for this case. Every algorithm was executed for 50 times, to normalize the variations that occur when executed every time. In table II, the results of the trial having minimum cost for three algorithms was mentioned. From table 2, it is observed that ABC performed well in reducing cost and violation compared to BGA.

TABLE II: Comparison of Two Algorithms for Case 1
(Trial Having Minimum Cost Among 50 Trials)

Parameters	ABC	BGA
P1 (MW)	432.518928	426.49071
P2 (MW)	301.65916	302.93255
P3 (MW)	131.70553	136.51026
P-Total (MW)	865.88362	865.93352
Cost ($/hr.)	**8344.62092**	8344.78384
Violation (MW)	$\mathbf{4.65672 \times 10^{-5}}$	0.01863
P-Loss (MW)	15.88367	15.95216

B. Case 2 – Six Units System

The six units' system was taken from [2], where all units are thermal and the load demand was 1263 MW, B coefficients are considered with 100MVA base capacity. The six-unit system data is mentioned in Table III and B- coefficients data was mentioned in Table IV.

TABLE III: 6 Units System Data

Unit Number	P_{min} (MW)	P_{max} (MW)	a	b	c
1	100	500	240	7.0	0.0070
2	50	200	200	10.0	0.0095
3	80	300	220	8.5	0.0090
4	50	150	200	11.0	0.0090
5	50	200	220	10.5	0.0080
6	50	120	190	12.0	0.0075

TABLE IV: B-Coefficients Data of 6 Unit System

0.0017	0.0012	0.0007	-0.0001	-0.0005	-0.0002
0.0012	0.0014	0.0009	0.0001	-0.0006	-0.0001
0.0007	0.0009	0.0031	0.0000	-0.0010	-0.0006
-0.0001	0.0001	0.0000	0.0024	-0.0006	-0.0008
-0.0005	-0.0006	-0.0010	-0.0006	0.0129	-0.0002
-0.0002	-0.0001	-0.0006	-0.0008	-0.0002	0.0150

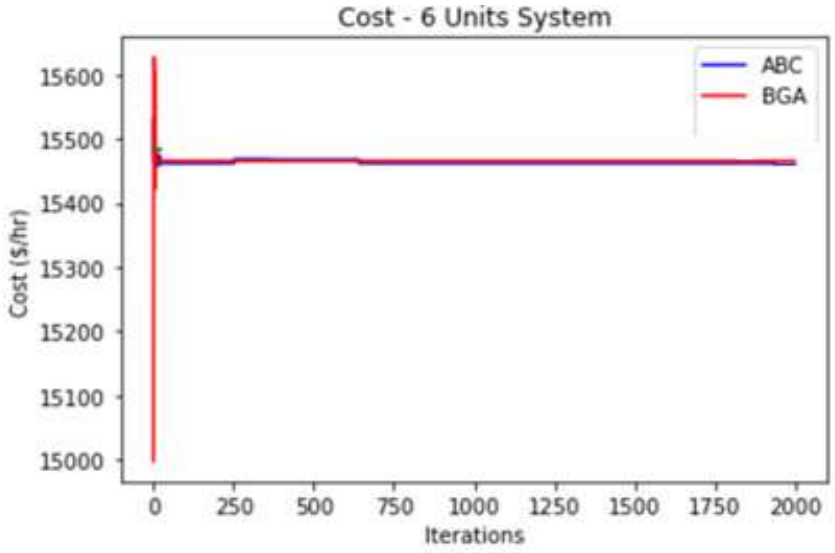

Fig. 4. Convergence of cost for 2 algorithms (case-2)

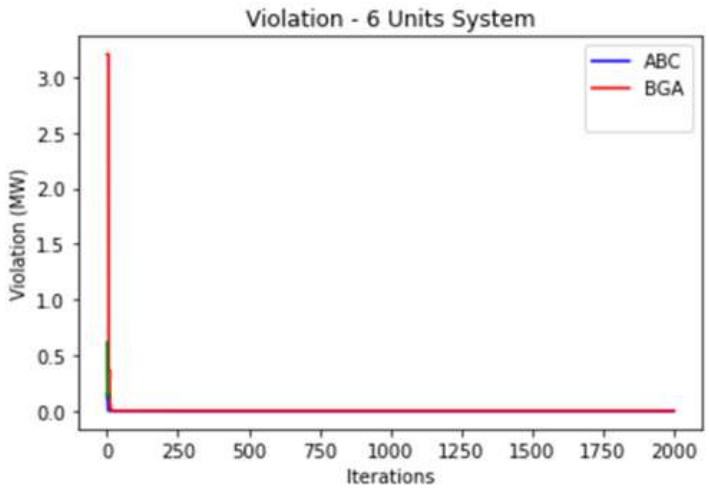

Fig. 5. Convergence of violation for 2 algorithms (Case-2)

TABLE V: Comparison of Two Algorithms for Case 2
(Trial Having Minimum Cost Among 50 Trials)

Parameters	ABC	BGA
P1 (MW)	442.45364	472.62952
P2 (MW)	184.45578	164.80938
P3 (MW)	267.50108	247.95698
P4 (MW)	138.55905	138.66080
P5 (MW)	166.38191	171.84750
P6 (MW)	76.10648	79.62854
P-Total (MW)	**1275.45797**	1275.53274
Cost ($/hr.)	**15445.21818**	15451.22916
Violation (MW)	$\mathbf{1.43313 \times 10^{-5}}$	0.00194
P-Loss (MW)	12.45799	12.53469

C. Case 3 - Fifteen Units System

The fifteen units' system was taken from [15], and the load demand was 2630 MW, B coefficients are considered with 100 MVA Base capacity. The six-unit system data was mentioned in table VI.

TABLE VI: 15 Units System Data

Unit-Number	P_{min} (MW)	P_{max} (MW)	a	b	c
1	150	455	671	10.1	0.000299
2	150	455	574	10.2	0.000183
3	20	130	374	8.8	0.001126
4	20	130	374	8.8	0.001126
5	150	470	461	10.4	0.000205
6	135	460	630	10.1	0.000301
7	135	465	548	9.8	0.000364
8	60	300	227	11.2	0.000338
9	25	1620	173	11.2	0.000807
10	25	160	175	10.7	0.001203
11	20	80	186	10.2	0.003586
12	20	80	230	9.9	0.005513
13	25	85	225	13.1	0.000371
14	15	55	309	12.1	0.001929
15	15	55	323	12.4	0.004447

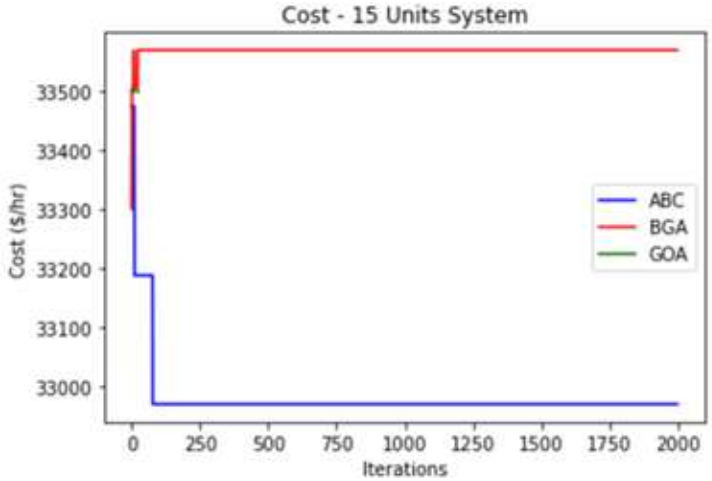

Fig. 6. Convergence of cost for 2 algorithms (Case-3)

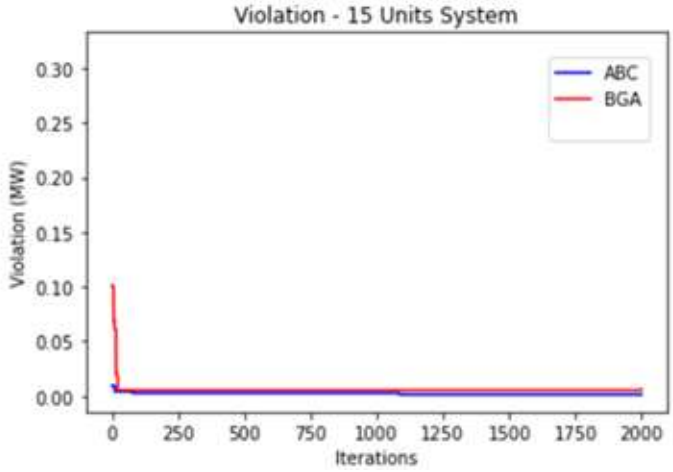

Fig. 7. Convergence of violation for 2 algorithms (Case-3)

TABLE IV: Comparison of Two Algorithms for Case 3

(Trial Having Minimum Cost Among 50 Trials)

Parameters	ABC	BGA
P1 (MW)	343.60601	415.64516
P2 (MW)	430.68879	388.51417
P3 (MW)	118.42544	123.22580
P4 (MW)	117.48871	90.21505
P5 (MW)	433.09797	209.43304
P6 (MW)	337.46807	420.28836
P7 (MW)	465	426.93548
P8 (MW)	60	146.56891
P9 (MW)	39.34941	94.77223
P10 (MW)	151.52428	125.29325
P11 (MW)	47.96969	52.60997
P12 (MW)	49.21817	75.77712
P13 (MW)	43.70004	26.99413
P14 (MW)	15	45.65493
P15 (MW)	15	23.44574
P-Total (MW)	2667.53663	2665.37341
Cost ($/hr.)	**32845.53768**	32969.13445
Violation (MW)	**0.00165**	0.00311
P-Loss (MW)	37.53828	35.37372

VII. Conclusion

Table V compares the average values of 50 trials of three algorithms for all three cases. By observing the table for case 1, the ABC algorithm minimized the cost than BGA, whereas ABC managed to balance the violation (270 Watts) and cost minimizing them optimally, similarly in case 2 and 3 ABC managed to minimize both cost and violation. ABC showed best results in all cases because of proper exploitation at time updating the food sources. BGA failed to minimize cost and violation because of dependency on best values at selection process, due to which an early convergence is seen.

TABLE V: Comparison of average values of 50 trials of two algorithms for all three cases

Number of Buses	Parameters	ABC	BGA
3 Unit System (case 1)	P-Total (MW)	865.80297	866.15835
	Cost ($/hr.)	**8348.47508**	8367.80694
	Violation (MW)	**0.00027**	0.01019
	P-Loss (MW)	15.80325	16.16854
6 Unit System (case 2)	P-Total (MW)	1275.60285	1276.00508
	Cost ($/hr.)	**15452.54211**	15483.17884
	Violation (MW)	**0.00020**	0.00617
	P-Loss (MW)	12.60306	13.01126
15 Unit System (case 3)	P-Total (MW)	2667.20480	2676.03550
	Cost ($/hr.)	**33019.04735**	33344.10255
	Violation (MW)	**0.00077**	0.00455
	P-Loss (MW)	37.20557	46.04005

References

[1] P. P. Biswas, "Evolutionary algorithms for solving power system optimization problems," Doctoral thesis, Nanyang Technological University, Singapore, 2019.

[2] Sheri Abhishek Reddy, M. Sailaja Kumari, A review of switching overvoltage modeling in UHV AC transmission lines, Electric Power Systems Research, Volume 236, 2024, 110902, ISSN 0378-7796, https://doi.org/10.1016/j.epsr.2024.110902.

[3] Sheri Abhishek Reddy, M. Sailaja Kumari, Modelling Suggestions for UHV AC Transmission Line using PSCAD/EMTDC, CIGRE India Journal, Volume 12, 2023, 28-32, ISSN 2250-0081.

[4] Sheri Abhishek Reddy, M. Sailaja Kumari, Mitigation of Switching Overvoltages in UHV AC Systems, CIGRE India Journal, Volume 8, 2023, 25-38, ISSN 2250-0081.

[5] Zwe-Lee Gaing, “Particle Swarm Optimization to Solving the Economic Dispatch Considering the Generator Constraints,” in IEEE Trans. on Power Syst., vol. 18, No. 3, pp. 1187-1195, Aug 2003.

[6] M. Sulaiman , "Implementation of Improved Grasshopper Optimization Algorithm to Solve Economic Load Dispatch Problems", Hacettepe Journal of Mathematics and Statistics, pp. 1-21, doi:10.15672/hujms.507579

[7] S. Hemamalini & Sishaj P. Simon (2010): Artificial Bee Colony Algorithm for Economic Load Dispatch Problem with Non-smooth Cost Functions, Electric Power Components and Systems, 38:7, 786-803.

[8] A. Rudolf and R. Bayrleithner, “A Genetic Algorithm for Solving the Unit Commitment Problem of a Hydro-Thermal Power System,” in IEEE Trans. On Power Systems, vol 14, No.4, pp. 1460-1468, Nov 1999.

[9] Aniruddha Bhattacharya and Pranab Kumar Chattopadhyay, “Biogeography-Based Optimization for Different Economic Load Dispatch Problems,” in IEEE Trans. On Power Systems, vol 25, No.2, pp. 1064-11077, May 2010.

[10] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Technical Report-TR06, Erciyes University, Kayseri, Turkey, 2005.

[11] D. Karaboga and B. Basturk, "A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," Journal of Global Optimization, vol. 39, no. 3, pp. 459-471, 2007.

[12] K. V. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization. Berlin, Germany: Springer, 2005.

[13] M. Dorigo and T. Stützle, Ant Colony Optimization. Cambridge, MA, USA: MIT Press, 2004.

[14] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.

[15] M. Mitchell, "An Introduction to Genetic Algorithms," MIT Press, 1998.