Asian Engineering Review
ISSN: 2409-6725
Vol. 1, No. 2, 26-35, 2014
http://asianonlinejournals.com/index.php/AER

Investigating the Effects of Evolutionary Parametric Tuning for Static Economic Load Dispatch Problems

***Sunny Orike*[1*]**

[1]Department of Computer Science, Heriot-Watt University, Edinburgh, United Kingdom

Abstract

This paper investigates the effects of evolutionary parametric tuning in realizing optimal solutions for static economic load dispatch (SELD) problems in the electrical power industry. During evolutionary algorithm (EA) implementation, design parameters need to be carefully chosen because of their contributory influence on the success and overall performance of the EA. The paper considers instances of the SELD problem involving minimum/maximum generation limits and power (load) balance. Simulation results are provided for novel intelligent mutation approaches from this investigation, and compared with reported results for other recent alternate algorithms in literature on two benchmark cases involving 6 and 20 generating units, where they exhibited superior performances in terms of lowest cost and meeting customers' demands.

Keywords: Economic load dispatch, Evolutionary algorithm, Fitness function, Optimization, Simulation, Smart mutation.

Contents

** Corresponding Author*

1. Introduction

When building EAs, there are various design decisions that need to be made. The encoding type, population size, crossover and mutation rates, etc, need to be carefully chosen, as their values have combined effects on the overall performances of EAs. This is called "tuning" of evolutionary parameters [1]. There is a "parameter space" that consists of all the possible sets of EA parameters. These are: population size, crossover rate and mutation rate. An element of this space is a triple defined as *(pop_size, cross_rate, mut_rate)*; e.g. (100, 0.8, 0.5) [2]. This varies widely between different problems and encoding, and it is very difficult to arrive at specific values that work for all EAs. De Jong determined systematically the effects of the different parameters on the performance of most EAs [2], and concluded that the optimal parameters were: population size of 50 to 150 individuals, one-point crossover of rate 0.6 (and above) and a mutation rate in the region of 0.001. Grefenstette evolved an optimal set of parameters for De Jong's EA, and discovered the best values as: (30, 0.95, 0.01) [3]. While these values were widely used and provided good basis for much earlier work, however, as the range of problems expanded, it become clear that no single set of parameters are universally optimal. Based on the work of Grefenstette, it was therefore suggested to begin an EA search by considering a population size of 30 individuals, crossover rate of 0.6 and a mutation rate of 0.01 [1]. The tuning of genetic parameters for the current work is based on this recommendation.

The Static Economic Load Dispatch (SELD) problems handle single load optimization period (typically, one hour duration), in which the variables (generator outputs) do not vary with time. This paper investigates the effects of these parameters in realizing optimal solutions for SELD problems in the electricity generation industry. It considers instances of the problems, involving legal generation limits and load balance.

The remaining sections of the paper are organized as follows: Section 2 describes the workflow and various processes of a basic evolutionary algorithm; section 3 presents the methodology adopted in this work, with the details of the parametric tuning, including experimental design, simulation results and discussion made in chapter 4. Chapter 5 concludes the paper by reflecting on the findings and identification of potential future work.

2. Basic Evolutionary Algorithm

EAs are popular heuristic methods for realizing solutions of real-life optimization problems. Loosely reflecting Darwin's evolutionary theory [2], EAs use selection, crossover and mutation operators to create environments where populations of individuals (chromosomes) compete with one another, and only the fittest move from the current generation to the subsequent ones. Realization of optimal solutions to optimization problems using conventional EA, herein referred to as Basic EA (BEA) involves the following steps:

1. *Encode the problem to be solved as a string of chromosomes*
2. *Generate an initial population of chromosomes;*
3. *Evaluate the fitness of each chromosome in the population;*
4. *Select parent chromosomes from the population to form breeding pair;*
5. *Perform crossover and mutation on selected parent chromosomes to form offspring;*
6. *Evolve (replace parent population with the offspring population);*
7. *If stopping criterion is not reached, go back to step 3.*

2.1. Encoding a Problem

EA works on a population of chromosomes. A chromosome is a string which represents a solution to a particular problem. It is an abstraction of the deoxyribonucleic acid (DNA) chromosome in biology, which is conceived as a string of letters in English language alphabets [4]. The exact position in a chromosome is called a *gene*, while the value which is present at that location is called an *allele*. The way in which alleles are represented in a chromosome is referred to as *encoding*. The most popular or classical encoding is the binary encoding (bit-string representation with values 0 or 1). Others include: permutation encoding, real-value encoding and tree encoding [2, 5]. This paper uses a real-value encoding, where allele values represent real power output of generators.

2.2. Generating Initial Population

The initial population of an EA is a set of potential/candidate solutions to the problem. There are many methods of generating the initial population of chromosomes. The common method is random generation. Here, the allele values are random numbers based on the defined boundaries - the lower and upper limits of each generator. While this approach is efficient and provides a population covering the feasible solution space, the entire initial population may also be infeasible. That is, subsequent generations may not be as the previous ones, resulting in good solutions to evolve slowly. The initial population could also be the output of another search algorithm in a situation where a hybrid approach is used.

2.3. Fitness Evaluation

One of the first steps in using an EA to solve a specific problem is to specify a fitness function. This calculates the quality of candidate (potential) solutions to that problem. The fitness evaluation process uses this fitness function to compute the quality of each solution. In analogy to biology, the chromosome is the genotype, while the solution it represents is the phenotype [4]. The computation takes several factors and objectives into account, including the cost minimization/maximization, penalty handling, resources utilization, run time, etc.

2.4. Selection of Parents

Selection mechanism is used to choose individuals (parents) from the initial population to go into the mating pool (breeding), form where offspring is generated. This is the basis of new population. There are several selection methods available for use in EA implementations, with each of them having their unique and distinct characteristic features. An ideal selection method should be simple, computational efficient and suited for parallel implementation [5, 6]. Tournament selection (the method used in this paper), satisfies the above criteria, and is a robust selection

method commonly used in most EA implementations [6]. The general motive behind all selection methods is to provide a selection pressure in favor of better solutions, and balancing exploration against exploitation.

The tournament selection algorithm chooses *t* individuals from the population (uniformly at random, with replacement), and the highly-fit of those individuals is the one returned as 'selected'. In the context of selecting a set of *N* parents, this process is repeated *N* times. A tournament selection parameter *t* is used, which is called the tournament size. A commonly used tournament size is two, and this selection method is referred to as "binary tournament selection". The following general work flow of a binary tournament selection:

(i) Choose two individuals at random from initial population;
(ii) Pick a random number, r (between 0 and 1);
(iii) If r < k (where k is a user-defined parameter, over 0.5 but less than 1), the fitter of the two individuals is selected as parent to go into the mating pool;
(iv) Else, the less-fit individual is selected;
(v) Return the two individuals to the original population.

To prevent losing the best individuals of the population from one generation to the next, elitism is applied. This technique copies the top *N%* of the present population on to the next generation (where *N* is the elitism rate). Tournament selection is known to overcome the weaknesses of other selection methods, by preventing the domination of highly-fit solutions, thereby wiping out all useful information which may be present in the less-fit ones. It also eliminates too weak solutions which will result in too slow convergence.

2.5. Breeding

This is also known as recombination [4], a process which follows selection where the selected chromosomes (parents) from the current population are recombined to form a successor population (children). This is to stimulate the mixing of genetic composition of the parents when they reproduce. It is expected that more highly-fit chromosomes will result from this process since selection is biased to favor chromosomes with higher fitness. Breeding is achieved by applying *genetic operators*, which generate new candidate solutions by using parts of existing candidate solutions (the selected parents). The common operators are *crossover* and *mutation*. Crossover typically combines parts of two parent solutions to form two child solutions; however in general, such a *recombination* operator can generate a child from more than two parents, and generate one or more children. In contrast, a mutation operator always involves a single parent, and the child is a 'mutation' of that parent – that is, it will typically be the same as the parent except for changes in a small number of its alleles. Genetic operators are invariably non-deterministic.

Crossover exchanges the genes (genetic composition) between two parents. It takes two parents and produces two children, with the children inheriting a mix of genes from the parents. In most EAs, crossover occurs with a high probability, called the crossover rate. Following selection of parents, a random number between 0 and 1 is generated which is compared to the crossover rate. If the crossover rate is lower than the number, no crossover occurs and the parents progress to the next phase unchanged. But if the crossover rate is greater or equal to the number, crossover is done. One-point crossover is the simplest crossover operator. Other alternatives which are generalization of the one-point crossover are: two-point and multi-point crossover operations. Another form, called uniform crossover [4], selects uniformly between the allele values of parents at each point to form children. Figure 1 shows a one-point crossover acting on a binary encoding.

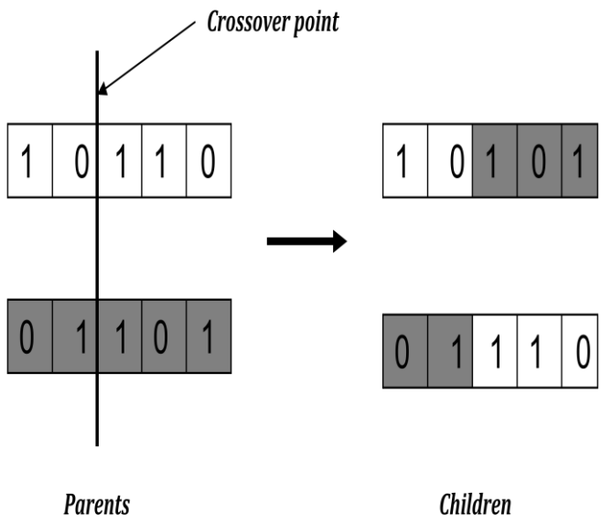

Figure-1. One-point crossover for binary encoding

Mutation is a major source of genetic variation. In an EA design, the mutation operator is applied to the resulting children solutions after the crossover, and allows new genetic patterns to be introduced, whether desirable or undesirable. Mutation is a standalone operator that is applied with its own 'mutation rate'. EAs need mutation to avoid genetic stagnation, because crossover cannot introduce new alleles to the population. Mutation usually occurs with a low probability. There are different mutation operators for the different types of encoding methods. For binary strings, it is implemented by randomly switching of bits, that is, 0 to 1 and 1 to 0. Figure 2 shows a one-point bit flipping mutation for a binary encoding.

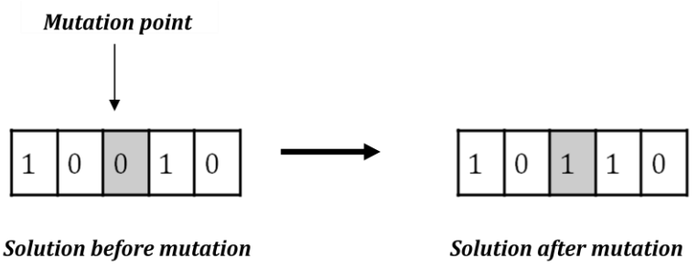

Figure-2. One-point, bit flipping mutation for binary encoding

For permutation strings, a mutation operator is designed such that the generated new strings fulfill the requirement of permutation encoding. Two elements in the string are randomly selected and swapped with each other, as shown in Figure 3, where "6" and "4" in the third and seventh positions are swapped with each other:

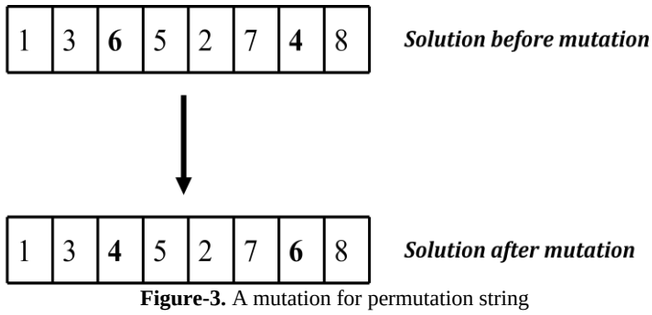

Figure-3. A mutation for permutation string

For value encoding string, the mutation operator randomly selects a number from the parent solution, changes its magnitude defined by the programmer, and returns the result in the child solution as shown in Figure 4 below, where the magnitude of the gene in the third position is changed from **12.99** to **13.57.**

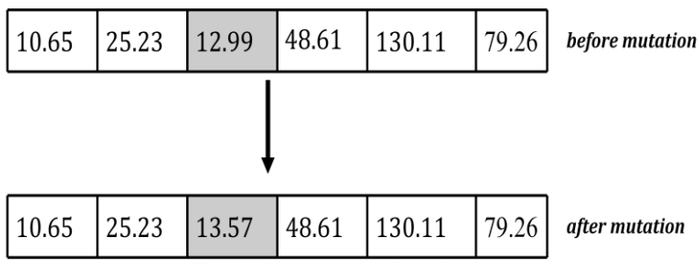

Figure-4. A mutation for value string

In its simplest form of operation, the genes undergoing mutation are 'randomly' selected, but in other cases, they could be 'targeted'. Other forms/types of mutation are: insertion, boundary, displacement, uniform mutation, Gaussian mutations [4]. In insertion mutation, a gene is selected at a random position and inserted also randomly into another position in the chromosome. Boundary randomly mutation replaces the gene with either lower or upper bound. In displacement mutation, a randomly selected portion of the chromosome is moved as a block to another location within the chromosome. Uniform mutation replaces the value of the selected gene with a uniform random value chosen between user-defined upper and lower bounds. Non-uniform mutation increases the probability that the amount of mutation will be close to zero with increased number of generation, Gaussian mutation adds a unit Gaussian distributed random value to the chosen gene.

2.6. Evolution

The resulting child population after mutation replaces the old population, forming the successor population. The selection and breeding processes are iterated, leading to a succession of 'generations' of solutions. At each next generation, the successor population becomes the source (parent) population. The EA goes through a number of generations until one or more of the stopping criteria (reaching a fixed number of iterations or finding a sufficiently good solution) are met, where the best chromosome in the resulting population is returned as the solution to the problem.

3. Methodology

The SELD problem is an optimization task whose goal is to find the optimal combination of online power generators that will minimize the total fuel cost to meet the total system's load demand while satisfying various equality and inequality constraints. This is done over an appropriate short-term period, usually one hour. For a thermal generating station, the unit fuel cost is shown in the quadratic form of:

$$Min\, F_c = \sum_{i=1}^{N} (a_i Pg_i^2 + b_i Pg_i + c) \tag{1}$$

Where F_c is the total generating cost (fuel cost in \$/h), Pg_i is the power output of unit i, a_i, b_i and ci are the fuel cost coefficients of unit i, N is the number of generating units. This is subject to the following generation and power balance constraints:

$$Pg_i^{\min} \le Pg_i \le Pg_i^{\max} \tag{2}$$

$$\sum_{i=1}^{N} Pg_i - (P_D + P_L) = 0 \quad (3)$$

P_D is the total power demand and P_L is the power loss, whose value is determined by means of the Kron's loss formula [6]. The generalized fitness function given by:

$$F_c = \sum_{i=1}^{N} f_i(Pg_i) + q_1 \sum_{i=1}^{N} (Pg_i - P_D - P_L)^2 \quad (4)$$

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 [7-9]. The rule defined in (2) ensures that outputs of the generators are within the legal minimum and maximum limits.

An intelligent approach of capturing gene-specific contributions to generation costs, and using this information to help target the mutation operator, is at the heart of what is referred to, in this paper as a 'smart evolutionary algorithm' (SEA). The method basically combines BEA with a smart mutation algorithm. The description and basic pseudo-code for SEA is shown in Orike and Corne [10]. Three SEAs (SEA1, SEA2 and SEA3), resulting from three distinct variants of the smart mutator were implemented. *SEA1* assumes full mutation and uses tournament selection to decide which gene to mutate. In *SEA2*, there is a mutation probability, and when it is met, a smart mutation is done; otherwise, a standard random mutation is done. *SEA3* extends *SEA2*, but the value of the mutation probability starts at 0, and gradually moves to 1 towards the maximum number of generations.

4. Experiments and Results

The tuning of the evolutionary parameters was done in respect of SEA1, SEA2 and SEA3 for the SELD problem for two test cases involving 6 and 20 generating units. The effects on generation cost and meeting customers' load demand were investigated, analyzed and compared with BEA and other contemporary approaches in the literature.

4.1. Case I: 6 Generating Units

In [6, 10-13], the SELD problem was solved for 6 thermal units. Using the data from this paper, experiments were carried out to tune the evolutionary parameters - crossover rate, population size, tournament size and mutation rate.

Table-1. Summary of results for different crossover rates, averaged over 30 runs of SEA1 algorithm

Crossover Rate	**0.6**	**0.7**	**0.8**	**0.9**
Av Cost ($/h)	769.59	**758.19**	779.28	774.97
Std Dev	15.41	12.25	12.78	18.04

Table-2. Summary of results for different population sizes, averaged over 30 runs of SEA1 algorithm

Pop Size	**10**	**20**	**30**	**40**	**50**	**100**	**150**
Av Cost ($/h)	758.19	755.66	756.01	757.55	748.07	**738.73**	740.21
Std Dev	12.25	15.30	18.66	12.82	11.02	9.28	12.36

Table-3. Summary of results for different tournament sizes, averaged over 30 runs of SEA1 algorithm

Tournament Size	**2**	**4**	**6**	**8**	**10**
Av Cost ($/h)	**738.73**	742.22	741.01	741.64	744.64
Std Dev	9.28	11.00	10.45	13.28	12.77

Table-4. Summary of results for different smart mutation rates, averaged over 30 runs of SEA2 algorithm

Smart Mutation Rate	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**
Av Cost ($/h)	757.83	759.09	758.23	761.37	752.58	**749.54**	760.61
Std Dev	15.07	15.62	14.56	13.92	15.33	**13.34**	13.83

Table-5. Best resources allocation for different crossover rates in a single run of SEA2 algorithm

Units	**Crossover Rate**			
	0.6	**0.7**	**0.8**	**0.9**
1	173.15	168.89	120.10	95.48
2	20.77	24.02	68.39	56.82
3	19.41	48.30	31.12	77.99
4	19.48	10.91	13.31	30.84
5	21.57	15.56	21.22	25.67
6	29.21	16.58	30.03	26.79
Total Gen (MW)	283.61	284.29	284.18	283.58
Total Cost ($/h)	776.22	**715.62**	765.32	743.40
Loss (MW)	0.21	0.89	0.78	0.18

Total load demand was set at 283.40 MW and the experimental data, including loss coefficient B-matrix can be retrieved from [6, 10]. Starting with SEA1, Tables I, II and III summarize the results of different values for crossover

rate, population size and tournament rate, averaged over 30 runs; while Table IV summarizes the results of different mutation rates for SEA2.

Their respective resources allocations for the best solutions in the entire set of runs are shown in Tables V, VI, VII and VIII. This is in terms of realizing the two main goals of the dispatch, namely: lower cost of generation, and meeting load demand.

Table-6. Best resources allocation for different population sizes in a single run of SEA1 algorithm

Units	**Population Size**						
	10	**20**	**30**	**40**	**50**	**100**	**150**
1	168.89	111.19	159.80	154.80	135.25	151.31	161.21
2	24.02	76.03	25.95	34.44	33.69	24.69	27.39
3	48.30	48.64	45.48	44.51	46.69	48.89	45.61
4	10.91	15.58	17.44	27.63	33.44	23.45	19.87
5	15.56	11.33	12.57	13.63	15.36	15.69	17.66
6	16.58	21.08	22.92	13.62	20.83	19.85	15.25
Total Gen (MW)	284.29	283.84	284.14	288.62	284.66	283.88	286.98
Total Cost ($/h)	715.62	730.56	713.66	724.54	720.60	**709.60**	719.72
Loss (MW)	0.89	0.44	0.74	5.22	1.26	0.48	3.58

Table-7. Best resources allocation for different tournament sizes in a single run of SEA1 algorithm

Units	**Tournament Size**				
	2	**4**	**6**	**8**	**10**
1	151.31	164.27	122.69	141.82	120.72
2	24.69	26.66	43.06	40.04	65.25
3	48.89	46.14	49.57	49.55	49.44
4	23.45	18.27	30.23	17.93	20.40
5	15.69	15.43	13.37	15.45	16.07
6	19.85	14.58	24.56	20.91	12.57
Total Gen (MW)	283.88	285.35	283.42	285.69	284.44
Total Cost ($/h)	**709.60**	711.74	713.65	719.00	715.55
Loss (MW)	0.48	1.95	0.02	2.29	1.04

Table-8. Best resources allocation for different smart mutation rates in a single run of SEA2 algorithm

Units	**Smart Mutation Rates**						
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**
1	128.96	137.99	151.71	152.54	133.95	168.99	150.30
2	51.11	29.72	22.55	34.07	47.37	30.54	34.34
3	49.49	49.95	44.18	34.97	41.67	40.75	36.45
4	12.73	11.54	20.18	22.78	17.48	11.68	33.87
5	26.51	25.51	16.84	19.01	21.62	15.76	12.98
6	17.44	29.86	28.33	20.55	21.36	15.71	15.76
Total Gen (MW)	286.23	284.58	283.79	283.92	283.44	283.44	283.69
Power Dem (MW)	283.40	283.40	283.40	283.40	283.40	283.40	283.40
Total Cost ($/h)	719.98	723.70	726.12	731.53	721.24	**711.86**	727.04
Loss (MW)	2.83	1.18	0.39	0.52	0.04	0.04	0.29

From these initial experiments, tuned values for population size, tournament size, crossover rate, and mutation rates were respectively determined from the lowest costs realized with the parameters' values, given in Table IX, alongside with a fixed number of generations of 100, penalty factor, q_1= 50000, scaling factor, α = 0.1, as in Sayah and Zehar [6], and elitism rate [14], which were used for the main experiment involving *SEA1*, *SEA2* and *SEA3*, with results shown in Table X, averaged over 30 runs.

Table-9. Experimental parameters and values for the 6-unit problem

Parameters	**Values**
Population size	100
Tournament size	2
Crossover rate	0.7
Smart mutation rate	0.6
No of Generations	100
Elitism Rate	10%
No. or runs	30

Table-10. Summary of results, for each of SEA1, SEA2 and SEA3 approaches on the 6-unit problem

	SEA1	SEA2	SEA3
Ave Cost ($/h)	738.73	749.54	744.49
Std Dev	13.74	13.34	10.57
Min Cost ($/h)	709.60	711.86	710.13
Max Cost ($/h)	759.20	766.97	758.97

To further demonstrate the efficiency of the SEAs, the distribution pattern of the best solutions in each of the 30 runs was made, as shown in Figure 5, which alongside Table X shows that the range of variation of the costs from each independent run is relatively small, and equally distributed between the minimum and maximum costs.

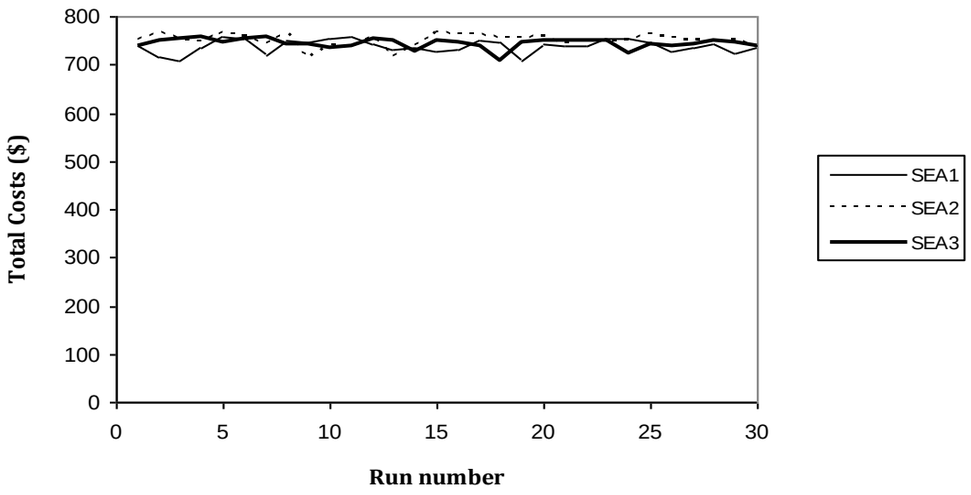

Figure-5. Distribution of generation costs for SEA1, SEA2 and SEA3, on the 6-unit problem

The results of SEA1, SEA2 and SEA3 were compared with BEA [10], Differential Evolution [6], Genetic Algorithm [6], Successive Linear Programming [12] and the Quasi-Newton Method [11]. Table XI summarizes the comparison results based on the best resources allocation to the units, which shows superior performance of the three SEAs in terms of both lower generation costs and lower power losses. All the three SEAs performed very well on this problem, having low total cost of $709.60/h, $711.86/h and $710.13/h respectively for SEA1, SEA2 and SEA3; and reduced power losses of 0.48MW, 0.048MW and 1.0MW respectively.

Table-11. Best resources allocation and comparison with other approaches on the 6-unit problem

Units	SEA1	SEA2	SEA3	BEA [10]	DE [6]	SLP [12]	GA [6]	QN [11]
1	151.31	168.99	130.44	171.58	177.51	175.25	179.37	170.24
2	24.69	30.54	41.12	49.26	48.61	48.34	44.24	44.95
3	48.89	40.75	49.63	22.63	20.91	21.21	24.61	28.90
4	23.45	11.68	27.55	21.20	21.64	23.60	19.90	17.48
5	15.69	15.76	19.45	12.73	12.47	12.25	10.71	12.17
6	19.85	15.71	61.21	14.17	12.02	12.33	14.09	18.47
Total Gen (MW)	283.88	283.44	284.40	292.11	293.16	292.98	292.92	292.21
Total Dem (MW)	283.40	283.40	283.40	283.40	283.40	283.40	283.40	283.40
Loss (MW)	0.48	0.04	1.00	8.71	9.76	9.58	9.52	8.81
Total Cost ($/h)	709.60	711.86	710.13	801.30	803.07	803.08	803.69	807.78

4.2. Case II: 20 Generating Units

In [10, 14, 15], the SELD problem was solved for 20 generating units, investigating the performance of SEA in larger problem case. The total load demand was 2500 MW, and the experimental data, can be retrieved from [10, 14, 15]. Several initial experiments were performed to select appropriate values for scaling factor (α) and load balance penalty factors (q_1). Starting with SEA1, and using the previously tuned values of the parameters from the experiments involving 6 units, Tables XII and XIII summarize the results of tuning q_1 and α, from where the values: 50,000 and 0.2 were selected respectively, average over 30 runs. Their respective best resources allocations in the entire runs are shown in Tables XIV and XV, in terms of lower cost of generation and meeting load demand.

Table-12. Summary of results for tuning power balance penalty factor (q_1), on the 20-unit problem for SEA1

	q_1						
	5	**10**	**50**	**500**	**5,000**	**50,000**	**500,000**
Ave Cost ($/h)	61552.11	61705.44	61782.95	61745.74	61600.49	**61108.99**	61710.03
Std Dev	667.18	497.58	771.90	795.51	647.75	547.51	740.68

Table-13. Summary of results for tuning scaling factor on the 20-unit problem for SEA1

	α				
	0.1	***0.2***	***0.3***	***0.4***	***0.5***
Ave Cost ($/h)	61569.85	61108.99	61470.32	61487.66	61664.60
Std Dev	807.29	547.51	570.23	672.05	679.05
	α				
	0.6	***0.7***	***0.8***	***0.9***	***1.0***
Ave Cost($/h)	61255.18	61371.29	61743.99	61406.48	61900.04
Std Dev	901.48	625.59	501.21	541.59	1097.01

Table14. Best resources allocation for diff. power balance penalty factor, on the 20-unit problem for SEA1

Units	**q_1**						
	5	**10**	**50**	**500**	**5,000**	**50,000**	**500,000**
1	541.87	318.62	488.71	390.97	307.89	212.08	386.69
2	191.59	155.34	65.41	88.84	125.65	75.33	101.72
3	100.73	80.44	112.53	69.68	108.83	159.53	70.46
4	151.07	198.61	96.77	73.51	101.43	153.53	78.34
5	107.58	140.25	76.50	80.94	64.88	145.30	126.46
6	39.25	98.89	53.54	21.59	36.07	31.89	59.64
7	120.19	107.13	91.99	104.19	76.19	41.08	101.40
8	68.31	81.04	149.31	146.89	129.49	142.45	61.10
9	146.19	162.19	130.33	122.25	119.11	74.34	127.23
10	82.06	33.78	72.61	130.62	148.31	117.10	52.56
11	131.99	207.13	221.71	207.83	277.63	281.14	226.03
12	197.05	210.50	388.51	431.27	469.72	413.08	451.76
13	83.52	154.80	82.43	152.44	109.93	96.47	125.94
14	125.21	110.62	98.31	43.36	96.66	127.99	80.99
15	116.68	117.95	159.69	142.64	118.09	79.41	172.67
16	49.59	48.05	24.31	35.50	28.30	40.04	45.38
17	46.94	46.62	33.79	52.43	32.63	53.01	43.51
18	61.74	76.90	35.79	87.36	35.63	110.82	44.70
19	53.56	72.57	71.09	60.75	60.41	70.10	95.52
20	86.08	83.21	52.49	70.59	55.11	75.77	54.99
Total Gen (MW)	2501.19	2504.64	2505.83	2505.65	2501.96	2500.22	2507.09
Total Dem (MW)	2500.00	2500.00	2500.00	2500.00	2500.00	2500.00	2500.00
Total Cost ($/h)	60801.21	60999.65	60692.22	60711.04	60720.35	**60522.37**	60580.86
Loss (MW)	1.19	4.64	5.83	5.65	1.96	**0.22**	7.09

Table-15. Best resources allocation for different scaling factor, on the 20-unit problem, for power balance penalty factor (q_1) = 50,000 for SEA1

Units	**α**					
	0.1	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
1	306.73	212.08	253.32	582.80	533.40	388.85
2	151.69	75.33	131.13	96.93	112.56	171.84
3	116.50	159.53	148.42	54.50	124.79	176.84
4	174.69	153.53	108.47	146.56	103.17	176.10
5	77.89	145.30	58.49	142.65	94.62	81.16
6	73.81	31.89	57.68	43.12	52.25	28.91
7	65.13	41.08	120.67	105.69	91.98	93.04
8	79.19	142.45	99.26	50.54	85.64	107.64
9	129.16	74.34	150.52	136.80	53.21	166.47
10	105.37	117.10	104.92	62.43	148.29	92.87
11	235.31	281.14	273.32	158.88	121.28	138.71
12	439.05	413.08	296.66	354.77	410.90	254.69
13	73.28	96.47	130.16	139.27	53.70	97.55
14	125.38	127.99	32.20	83.78	96.33	121.59
15	61.55	79.41	122.51	46.42	116.21	98.93
16	57.53	40.04	74.00	29.39	47.11	45.91
17	39.44	53.01	81.92	38.52	59.62	75.61
18	68.59	110.82	99.23	118.97	66.54	76.74
19	52.38	70.10	81.77	47.14	78.88	52.61
20	20.10	75.77	79.32	65.61	57.07	53.99
Total Gen (MW)	2501.78	2500.22	2503.97	2504.77	2507.54	2500.05
Total Dem (MW)	2500.00	2500.00	2500.00	2500.00	2500.00	2500.00
Total Cost ($/h)	60837.01	60522.37	61054.07	60576.81	60850.22	61028.95
Loss (MW)	1.78	0.22	3.97	4.77	7.54	0.05

The tuned values α, q_1, and those of tournament size, crossover rate, mutation rate from experiments involving 6 units; including the same population size and number of generations from [14, 15] (for uniformity of equal number of fitness evaluations) are as shown in Table XVI, which were used for the main experiment, with results shown in Table XVII and Figure 6, average over 30 runs.

Table-16. Experimental parameters and values for the 20-unit problem

Parameters	Values
Population size	30
Tournament size	2
Crossover rate	0.7
Smart mutation rate	0.6
No of Generations	100
Elitism Rate	10%
α	0.2
q_1	50,000

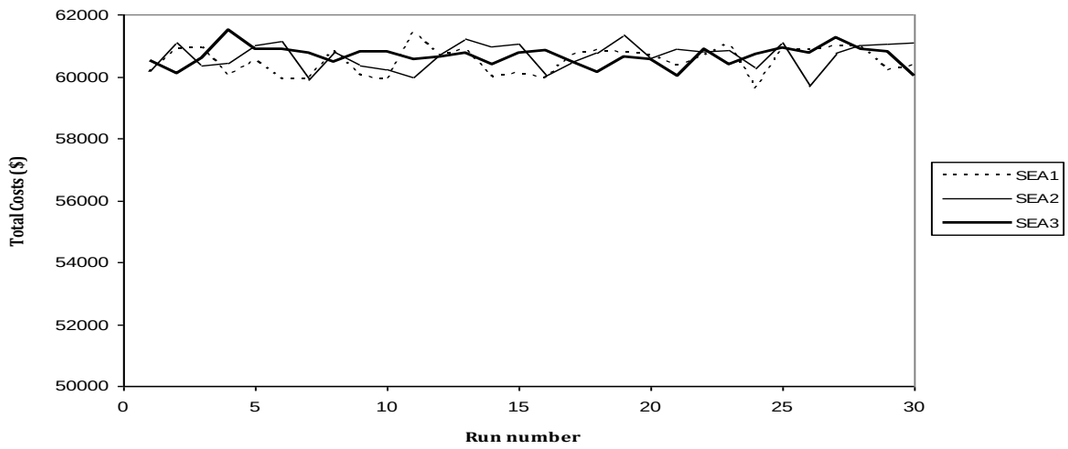

Figure-6. Distribution of costs for SEA1, SEA2 and SEA3 on the 20-unit problem

Table-17. Summary of results, and comparison with other approaches on the 20-unit problem

	SEA1	SEA2	SEA3	BEA [10]	PSO [15]	LIM [14]	HNN [14]
Av Cost ($/h)	60,492.99	60,671.28	60,659.80	61,654.76	61,171.84	-	-
Std Dev	464.65	442.11	331.24	515.57	532.44	-	-
Max Cost ($/h)	61,422.78	61,359.72	61,498.31	62,648.52	63,184.63	-	-
Min Cost ($/h)	59,588.38	59,687.80	60,003.02	60,727.19	60,760.25	62,456.64	62,456.63

Table-18. Best resources allocation and comparison with other approaches on the 20-unit problem

Units	SEA1	SEA2	SEA3	BEA [10]	PSO [15]	LIM [14]	HNN [14]
1	289.17	482.25	598.68	520.89	536.32	512.7805	512.7804
2	143.69	186.31	90.90	115.14	106.56	169.1033	169.1035
3	139.32	77.08	63.61	122.24	98.71	126.8898	126.8897
4	53.29	88.32	87.35	94.36	117.32	102.8657	102.8656
5	110.31	100.97	113.77	86.18	67.08	113.6836	113.6836
6	93.32	23.15	46.52	60.39	51.47	73.5710	73.5709
7	121.79	97.58	118.37	56.63	47.73	115.2878	115.2876
8	116.38	122.65	120.57	87.13	82.43	116.3994	116.3994
9	89.77	154.67	74.99	49.45	52.09	100.4062	100.4063
10	125.11	111.42	88.70	114.46	106.51	106.0267	106.0267
11	255.72	159.23	215.88	200.39	197.94	150.2394	150.2395
12	315.95	367.03	233.96	410.74	488.33	292.7648	292.7647
13	65.91	99.12	153.89	117.98	99.95	119.1154	119.1155
14	112.47	49.61	84.57	65.63	79.89	30.8340	30.8342
15	80.27	85.92	132.17	125.34	101.53	115.8057	115.8056
16	38.32	52.92	36.65	33.21	25.84	36.2545	36.2545
17	58.66	69.92	43.61	82.43	70.02	66.8590	66.8590
18	94.51	52.46	41.88	60.43	53.95	87.9720	87.9720
19	117.02	46.76	94.70	71.28	65.43	100.8033	100.8033
20	79.06	73.70	59.23	34.01	36.26	54.3050	54.3050
Total Gen (MW)	2500.02	2501.06	2500.00	2508.31	2512.33	2591.9671	2591.97
Total Dem (MW)	2500.00	2500.00	2500.00	2500.00	2500.00	2500.00	2500.00
Total Cost ($/h)	60846.10	60609.72	60483.14	60727.19	60760.25	62456.64	62456.63
Loss (MW)	0.02	1.06	0.00	8.31	12.33	91.97	91.97

Simulation of SEA1, SEA2 and SEA3 were compared with BEA [10], Particle Swarm Optimization (PSO) [15], Lambda-Iteration Method (LIM) [14] and Hopfield Neural Network (HNN) [14], all with the same set of data.

Tables XVII and XVIII summarize the comparison results, including typical resources allocation. The distribution curve of the best solutions in each of the 30 runs as shown in Figure 6, alongside Table XVII shows that the range of variation of the total costs from each run is relatively smallest in SEA3. From Table XVII, SEA1 has both lowest average cost and lowest minimum cost. Generally, the results of SEA1, SEA2 and SEA3 were better than those of the other approaches. In Table XVIII, SEA3 has the lowest generation cost of $60483/h; against $60609.72/h, $60727.19/h, $60760.25/h, $60846.10/h, $62456.63/h and $62456.64/h respectively for SEA2, BEA, PSO, SEA1, LIM and HNN. No power was lost in SEA3, as it exactly generated the customers' load demand of 2500MW; against 0.02MW, 1.06MW, 8.3139MW, 12.33MW, in SEA1, SEA2, BEA, PSO respectively, and 91.97MW in both LIM and HNN. From the above results, SEA1 and SEA3 appear to be the best optimizing approach in this test case. As noted in Table XVIII, the generation cost of $60,483/h from the resources allocation of SEA3 is the best ever seen in the literature to date for this problem, in terms of lowest cost and meeting load demand, with no power loss.

5. Conclusion

The paper performed rigorous tuning of evolutionary parameters to select values for experimental runs in SELD problems on two benchmark cases involving 6 and 20 generating units, but with a focus on the larger case. Simulation results of three novel EAs (SEA1, SEA2 and SEA3) were compared with those reported for a range of recent alternative algorithms, where they exhibited superior performances. As prospective areas of future work in this optimization problem, there are many opportunities to investigate variations in the general approach to the algorithm. When using a smart mutator, it is possible that in some problems, the chosen genes will cycle, e.g. mutating gene 1, moves the problem to gene 3, and mutating gene 3 next time moves the problem moves back to gene 1, and so on. The current work focused on performance in the application domain, and did not investigate this issue in depth. Future work could attempt a detailed investigation of this phenomenon, leading to designs for new, adaptive approaches to smart mutation that could avoid this situation. A record could also be kept of which genes that have been involved in smart mutation. This could be a variant of *SEA1*, such that when doing the tournament selection, a generator that was involved in smart mutation in the previous generation should not allowed to be selected.

References

[1] A. Rangel-Merino, J. L. López-Bonilla, and R. Linares y Miranda, "Optimisation method based on genetic algorithms," *Apeiron, Roy Keys Inc*, vol. 12, pp. 393–408, 2005.

[2] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc, 1989.

[3] J. Cogley, "Designing implementing and optimising an object-oriented chess system using a genetic algorithm in java and its critical evaluation," Research Report, The Open University, 2001.

[4] J. McCall, "Genetic algorithms for modelling and optimisation," *Journal of Computational and Applied Mathematics, Elsevier*, vol. 184, pp. 205 – 222, 2005.

[5] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, pp. 193 – 212, 1995.

[6] S. Sayah and K. Zehar, "Using evolutionary computation to solve the economic load dispatch problem," *Leonardo Journal of Sciences*, vol. 12, pp. 67 – 78, 2008.

[7] V. R. Pandi and B. K. Panigrahi, "Dynamic economic load dispatch using hybrid swarm intelligence based harmony search algorithm," *Journal of Expert Systems with Applications*, vol. 38, pp. 8509 – 8514, 2011.

[8] R. Balamurugan and S. Subramanian, "Differential evolution-based dynamic economic dispatch of generating units with valve-point effects," *Electrical Power Components and Systems*, vol. 36, pp. 828 – 843, 2008.

[9] S. Orike and D. W. Corne, "An evolutionary algorithm for bid-based dynamic economic load dispatch in a deregulated electricity market," presented at the Y. Jin and S. A. Thomas (Eds.). 13th IEEE UK Workshop on Computational Intelligence (UKCI 2013), University of Surrey, Guildford, 9th – 11th September 2013, 2013.

[10] S. Orike and D. W. Corne, "Improved evolutionary algorithms for economic load dispatch optimisation problems," in *Proceedings of 12th IEEE UK Workshop on Computational Intelligence (UKCI), Edinburgh*, 2012.

[11] T. Bouktir, L. Slimani, and M. Belkacemi, "A genetic algorithm for solving the optimal power flow problem," *Leonardo Journal of Sciences*, vol. 4, pp. 44 – 58, 2004.

[12] S. Sayah, K. Zehar, and N. Bellaouel, "A successive linear programming based method for solving optimal power flow problems," in *Proceedings of the 1st International Meeting on Electronics and Electrical Science and Engineering, University of Djelfa, Algeria*, 2006.

[13] S. Orike and D. W. Corne, "Constrained elitist genetic algorithm for economic load dispatch: Case study on Nigerian power system," *International Journal of Computer Applications, Foundation of Computer Science, New York, USA*, vol. 76, pp. 27 – 33, 2013.

[14] C. T. Su and C. T. Lin, "New approach with a hopfield modelling framework to economic dispatch," *IEEE Transactions on Power Systems*, vol. 15, pp. 541 – 545, 2000.

[15] L. S. Coelho and C. Lee, "Solving economic load dispatch problems in power systems using chaotic and gaussian particle swarm optimisation approaches," *International Journal of Electrical Power & Energy Systems*, vol. 30, pp. 297 – 307, 2008.