CHAPTER 2

REVIEW OF EVOLUTIONARY COMPUTATION ALGORITHMS

2.1 INTRODUCTION

The first issue of the IEEE Transactions on Evolutionary Computation appeared in April 1997. This signifies the beginning of the acceptance of evolutionary computation as a tool for optimization by researchers worldwide. Although the origins of evolutionary computation can be traced back to 1950's, its application to power system economic operation problems received wide attention only in the 1990's.

Evolutionary computation is emerging as an efficient approach for solving difficult optimization problems and is based on the models of biological evolution. Evolutionary algorithms move towards the global optimum based on the fitness of the trial solutions or on individuals similar to the survival of the fittest individuals in biological evolution. After a number of generations the algorithm evolves to the best individual in a population which represents the optimum solution. In the field of evolutionary computation there are three evolutionary algorithms namely, evolutionary programming, evolutionary strategy and genetic algorithms.
2.2 EVOLUTIONARY PROGRAMMING

Evolutionary programming (Yang et al, 1996) searches for the optimal solution by evolving a population of candidate solutions over a number of generations or iterations. The evolution of solutions is carried out through mutation by Gaussian distribution and competitive selection. The schematic diagram of the evolutionary programming algorithm is shown in Figure 2.1. The general scheme of the evolutionary programming for optimization is briefly given below:

In case of vector representation of evolutionary programming approach, the real-valued decision variables to be determined are represented as a trial n-dimensional vector. Each vector is an individual of the population to be evolved. For economic dispatch problem, each vector is an array of generator loadings.

In some cases, matrix representation is found to be more convenient than vector representation. For example, in short term generation scheduling, each individual may be represented as a matrix with dimension equal to the number of generators by the number of scheduling periods and in transportation problems each individual is a matrix of order equal to number of sources by number of destinations.

The major steps involved in evolutionary programming approach are discussed as follows:
Figure 2.1 Schematic diagram of evolutionary programming algorithm
i. **Initialization**

An initial population of parent individuals $p_i$, $i = 1, 2, \ldots, K$ is generated randomly within a feasible range in each dimension and the distribution of initial trial parents is uniform.

ii. **Creation of offspring (mutation)**

Each parent vector $p_i$ generates an offspring vector by adding a Gaussian random variable with zero mean and pre-selected standard deviation to each individual of $p_i$. The Gaussian relationship between each parent and offspring guarantees that all possible combinations of decision variables within the domain can be generated. The $K$ parents create $K$ offsprings, thus resulting in $2K$ individuals in the competing pool.

iii. **Competition and selection**

Each individual in the competing pool is evaluated for its fitness. All individuals compete with each other for probabilistic selection. The first $K$ individuals with minimum fitness values (for a minimization problem) are retained to be parents of the next generation.

This process of creating offspring and selecting those with minimum fitness are repeated until there is no appreciable improvement in the minimum fitness value.
2.3 EVOLUTIONARY STRATEGY

Evolutionary strategy (Lee and Yang, 1998) is very similar to evolutionary programming approach. The difference is in the mutation process in which the offspring is generated by adding a random variable with zero mean and a constant standard deviation whose value depends on the size of the decision variables. In this algorithm, the constant standard deviation makes the procedure slow to converge towards optimal solutions.

In the competition process, the fitness of individuals of population size $2K$ are sorted in a descending order. The first $K$ individuals are kept as parents for the next mutation process.

2.4 GENETIC ALGORITHMS

Genetic Algorithm (GA) (Goldberg, 1989) is a popular approach among the evolutionary computation algorithms. It starts with a population of randomly generated candidate solutions and evolve towards better solutions by applying genetic operators such as selection, crossover, and mutation. The schematic diagram of a simple GA is shown in Figure 2.2.
Figure 2.2 Schematic diagram of simple genetic algorithm
In Genetic algorithms, a population of appropriate representations of candidate trial solutions is maintained. Each individual of the population represents a search point in the space of the potential solutions. The representation may be bit strings or floating point numbers. In GA, each candidate solution is termed as a string or a chromosome. With the initial population produced and evaluated, a new population is evolved by employing the three genetic operators as discussed below (Goldberg, 1991).

i. **Reproduction or selection**: The selection of strings (or trial solutions) to reproduce is on the basis of their relative fitness. This fitness proportionate selection is the roulette wheel selection, which means that chromosomes with a higher fitness value have a higher probability of contributing one or more offsprings in the next generation. This process of selection and copying is repeated until the size of the mating pool is the same as the population size.

ii. **Crossover**: In a simple GA, crossover proceeds in two steps:
   a. Two individuals are chosen as parents from the mating pool and
   b. A position in the bit strings is randomly determined as the crossover point.

   The offspring is generated by concatenating the left substring of one parent and the right substring of the other parent. This crossover operator is applied with a certain probability and it promotes exploration of new regions in the search space.

iii. **Mutation**: In this, random bits of the offspring are altered from a '1' to a '0' and vice versa. It is applied with a small probability. Mutation is needed because even though reproduction and
crossover effectively search for the optimum, occasionally they may become over-zealous and lose some potentially useful genetic material (1's or 0's at particular locations). Hence this mutation operator protects against such irrecoverable loss. It is like an insurance against premature loss of important genetic material. Crossover forces convergence while mutation forces diversity in the population.

The application of these three operators on parents in one generation produces offsprings. These offsprings replace the parents and the cycle of operations is repeated until the best solution is attained.

The following improvements have been introduced to improve the performance of GA depending upon the problem chosen.

i. Elitism: It ensures that best individuals are never lost in moving from one generation to the next.

ii. Crossover: Crossover strategies such as two point crossover, multi-point crossover and uniform crossover are applied.

iii. Adopting non-uniform mutation for fine local tuning: As the population ages bits located further to the right of a string get higher probability of being mutated, while those on the left have lesser probability. Such a mutation causes global search of the search space at the beginning and a local exploitation later on.
2.5 SUITABILITY OF EVOLUTIONARY COMPUTATION

The evolutionary computation can be applied to problems where heuristic solutions are not available or generally lead to unsatisfactory results or available with more complexity. For example it can be applied to problems such as:

i. Large scale non-linear optimization problems: This is due to its inherently parallel and robust performance. Dynamic programming approach can also be applied for these optimization problems. But it suffers from the curse of dimensionality.

ii. Problems which have numerous local optima: The traditional optimization methods converge to a local optimal solution while evolutionary algorithms converge to the global optimum due to its global search characteristics.

iii. Problems with non-smooth, non-continuous and non-differentiable functions: Evolutionary algorithms can be applied directly because it needs only the fitness function to guide its search direction and not its derivative or any other auxiliary information.

For the above mentioned types of optimization problems, evolutionary algorithms consistently outperform both the gradient techniques and other forms of random search techniques.

Of the three evolutionary algorithms, evolutionary programming and simple genetic algorithms are widely used for power system optimization problems. Hence, a comparison has been made between these two algorithms
and the reasons for selecting the evolutionary programming approach for solving the different kinds of economic dispatch problems are given below:

i. In simple GA, the solution space is discrete in nature and hence it is difficult to efficiently apply GA to an optimization problem in a continuous multi-dimensional space, which is frequently needed for practical optimization problems. If used, they must sacrifice precision with an increase in domain size, for a given fixed binary length. Whereas in evolutionary programming, floating point representation is invariably used for the decision variables of continuous variations and hence it is capable of representing quite large domains.

ii. In simple GA, the crossover and mutation operators are well defined for binary representation of chromosomes. But these operators are quite different when they work on floating point numbers. They have to be designed to preserve the constraints. Hence evolutionary programming is preferred where the only operation required is mutation which is well defined. By avoiding crossover between parents when creating offsprings, the individuality of each parent is respected.

iii. In simple GA, the offspring evolved through the process of selection, crossover and mutation forms a new population. The old population is completely replaced with new population. By doing so, the progress towards the solution may lose the advantages obtained in the last generation, in case the old population is better (more fit) than the new population. Due to
this fact the convergence is not smooth. Whereas in evolutionary programming, competition in the combined pool of search space comprising both parents and offsprings and the subsequent stochastic selection produce a smooth convergence.

iv. In GAs different settings of the parameters - population size, crossover and mutation probabilities do not yield identical solutions. The number of generations needed also varies significantly with the random number starting seed used. Hence the optimum tuning of these parameter settings is difficult to determine or it is necessary to take the average of several runs to determine the optimal solution. Whereas in evolutionary programming, the only two parameters to be tuned are the population size and the scaling factor which yield almost the same solution over a wide range of values. Hence the optimal or near optimal solution may be found in a fewer number of iterations.

iv. In GA, premature convergence is another important concern. This occurs when the population of chromosomes reaches a configuration such that crossover no longer produces offspring that can outperform their parents, as must be the case in a homogeneous population. Under such circumstances, all standard forms of crossover simply regenerate the current parents. Any further optimization relies solely on bit mutation and can be quite slow. Such a problem does not arise in evolutionary programming.

Due to these reasons, evolutionary programming has been preferred over GA for optimization.
In this chapter, the three evolutionary computation algorithms namely, evolutionary programming, genetic algorithms, and evolutionary strategy has been reviewed. A comparison between the first two has been made which highlights the superiority of evolutionary programming over GA thus making the evolutionary programming approach the preferred choice for power system economic dispatch problems.