CHAPTER 2

REVIEW OF EVOLUTIONARY COMPUTING TECHNIQUES

Introduction

Genetic Algorithms (GAs)
Evolution Strategy (ES)
Genetic Programming (GP)
Simulated Annealing (SA)
Tabu Search (TS)
Particle Swarm Optimization (PSO)
Evolutionary Programming (EP)

Conclusion
CHAPTER 2

REVIEW OF EVOLUTIONARY COMPUTING TECHNIQUES

2.1 INTRODUCTION

During the last thirty years there has been a growing interest in problem solving systems based on principles of evolution and hereditary. Evolution is certainly the unifying principle of modern biology. However, there are several opinions as to "what is being evolved?" Genetic algorithm models are driven by the perspective that "evolution is a process that operates on chromosomes rather than on the living beings they encode". In contrast, Evolutionary algorithms (Evolutionary programming and Evolution strategy) endorse the claim that "evolution is change in the adaptation and in the diversity of population of organisms". Evolutionary computation is the standard term that encompasses all these biologically motivated techniques. The emergence of massively parallel computers made these algorithms of practical interest. The best-known algorithms in this class include Swarm Intelligence (SI), Genetic algorithms (GAs), Evolution strategy (ES), Evolutionary programming (EP), Genetic programming (GP), Simulated annealing (SA), Tabu search (TS) and Particle Swarm Optimization (PSO).

2.2 GENETIC ALGORITHMS (GAs)

2.2.1 Overview

Genetic algorithms are adaptive heuristic search algorithms premised on the evolutionary ideas of natural selection and genetics. The basic concepts of GAs are designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem.

First pioneered by John Holland in 1960, Genetic algorithms have been widely studied, experimented and applied in many fields in engineering world. Not
only does GAs provide alternate methods for solving problems, it consistently outperforms other traditional methods. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GAs. They can solve problems that do not have a precisely defined solving methods, or if they do, when following the exact solving method would take far too much time. The genetic algorithm on the other hand, works only with the objective function information in a search space for an optimal parameter set. G.B. Sheble et al have solved the unit commitment problem and economic dispatch with valve point loading by Genetic algorithm method. K.P. Wang et al have solved the large scale economic dispatch problem by Genetic algorithm method and Chuan Ping Cheng et al have integrated Genetic algorithm with Lagrangian relaxation method to form a hybrid LRGA method for solving the unit commitment problem. They proved the effectiveness of the hybrid LRGA method by comparing it with Genetic algorithm method and Lagrangian relaxation method.

2.2.2 Components of Genetic Algorithm

GAs are derived from a simple model of population genetics. They have following five components:

(i) String representation of the control variables.

(ii) An initial population of strings.

(iii) An evaluation function that plays the role of the environment, rating the strings in terms of their fitness that is their ability to survive.

(iv) Genetic operators determine the composition of a new population generated from the previous one by reproduction, crossover and mutation.

(v) Value of the parameters that the GAs use.

Since GAs are based on natural genetics, there exist strong analogies between genetic algorithm and natural genetics. The strings are similar to chromosomes in biological systems, where the chromosomes are composed of
genes, which may take any of several forms called "alleles". If the control variables are represented in binary bits and concatenated to form a string, then it is called as binary coded GA and if the control variables are represented in real numbers, then it is called as real coded GA. GAs do not work with a single string but with a population of strings, which evolves iteratively by generating new strings taking the place of their parents. To begin, the initial population is generated randomly. The performance of each string is evaluated according to its fitness. Fitness is defined as a non-negative figure of merit to be maximized. It is associated directly with the objective function value (f) in the optimization. GAs treats the problem, as a black box in which the input is the strings and the output is their fitness. Because GAs proceeds only according to the fitness of the strings and not with other information, the properties of the fitness influence the GAs performance.

Three basic operators comprise a GA. They are reproduction, crossover and mutation. Reproduction is the mechanism by which the most highly fit members in a population are selected to pass on information to the next population of members. It effectively selects the fittest of the strings in the current population to be used in generating the next population. In this way, relevant information concerning the fitness of a string is passed along to successive generations. It can be shown that GAs actually allocate exponentially increasing trials to the most fit of these strings. Crossover serves as a mechanism by which strings can exchange information, possibly creating more highly fit strings in the process and allowing the exploration of new regions of the search space. There are many types of crossovers are available like single point crossover, multipoint crossover, uniform crossover and window crossover. The last of the GA operators is mutation, and is generally considered as a secondary operator. Mutation ensures that a string position will never be fixed at a certain value for all time. Like other stochastic methods, GAs requires a number of parameters, which are population size, probability of crossover, probability of mutation. Usually small population size, high crossover probability and low mutation probability are recommended.
2.2.3 Comparison of GAs with Conventional methods

The GAs can be distinguished from other optimization methods in the four different ways as follows:

(i) GAs use objective function information to guide the search, not derivatives or other auxiliary information.

(ii) GAs use a coding of the parameters used to calculate the objective function in guiding the search, not the parameter themselves.

(iii) GAs search through many points in the solution space at one time, not a single point.

(iv) GAs use probabilistic rules, not deterministic rules, in moving from one set of solutions (a population) to the next.

2.3 EVOLUTION STRATEGY (ES)

In 1963, Ingo Rechenberg and Hans-Paul Schwefel the students at the technical university of Berlin met and were soon to collaborate on experiments which used the wind tunnel of the Institute of flow engineering. During the search for the optimal shapes of bodies in a flow, they hit upon the idea of trying random changes in the parameters defining the shape, following the example of natural mutations. The Evolution strategy was born. D.B. Fogel remarks that "evolution can be categorized by several levels of hierarchy: the gene, the chromosome, the individual, the species, and the ecosystem". Thus, while GAs stress models of genetic operators, ES emphasizes mutational transformation that maintains behavioral linkage between each parent and its offspring at the level of the individual. The first applications of ES were experimental and addressed some optimization problems in hydro dynamics.

2.4 GENETIC PROGRAMMING (GP)

Genetic programming is an optimization technique based on the concept of Darwinian evolution. A population of individuals, each representing a potential
solution to the problem to be optimized, undergoes a process analogous to biological evolution in order to derive an optimal or at least near optimal solution. The solution offered by each individual is assigned fitness, a single numerical value that indicates how well that solution performs. New individuals are generated by procedures analogous to biological reproduction, with parents chosen from the existing population with a probability proportional to their fitness. The new individuals may replace less fit members of the population, so the overall population fitness improves with each generation. Individuals store their potential solutions as a collection of genes. In a GA, these may be arrays of bits, integers, floating functions, mathematical operators, variables or numerical constants. An individual's total collection of genes is called as genome.

A GP is an application of the GA approach designed to perform an automatic derivation of equations, logical rules or program functions. Rather than representing the solution to the problem as a string of parameters as in a conventional GA, the GP uses tree structure, the leaves of the tree or terminals represent input variables or numeric constants. Their values are passed to nodes, at the junctions of branches in the tree, which perform some arithmetic or program function before passing on the result further towards the root of the tree. Mutations are performed by selecting a parent and either modifying the value or variable returned by a terminal or changing the operation performed by a node. Crossover is performed by selecting two parents and grafting sub-trees at randomly selected nodes within their trees. The new individuals so generated again replace less-fit members of the population.

John Koza described the first implementation of GP, in LISP, in 1992. More recently, the GP method has been implemented in C / C++, making it more portable between computer platforms.

2.5 SIMULATED ANNEALING (SA)

The simulated annealing is an algorithm, which exploits the resemblance between the annealing of a metal and a minimization process. A metal is a system of many atoms. The total internal energy of the metal depends on its state in the forms
of relative position, orientation and motion of the atoms in the metal. Although the prediction of a specific state is almost impossible due to the extremely rapid microscopic movement of the atoms, the statistical properties of many replica systems of atoms in their (thermodynamic) equilibrium can be characterized. It has been observed that when a metal is annealed, or cooled slowly, the energy of the metal tends to assume the globally minimal value. Motivated by this observation, simulated annealing generates feasible solutions of a minimization problem which correspond to the states of a metal, with the cost of a feasible solution corresponding to the energy of the metal in a state. By moving among the feasible solutions the way the states of a metal under annealing would evolve, the global optimum of the problem can be approached with high probabilities. Simulated annealing has been successfully applied to many difficult combinatorial optimization problems. The method assumes no special problem structure and is highly flexible with respect to various constraints. R.E. Burkard et al have demonstrated the Simulated annealing method by applying it for the solution of combinatorial optimization problems. F. Zhuang et al have solved the unit commitment problem by simulated annealing and listed the key issues regarding SA. A.H. Mantawy et al have integrated SA with GA for solving the unit commitment problem and they demonstrated the effectiveness of hybrid algorithms over conventional GA and SA.

2.6 TABU SEARCH (TS)

2.6.1 Overview

Tabu search (TS) is a higher-level heuristic algorithm for solving combinatorial optimization problems. It is an iterative procedure that starts from any initial solution and attempts to determine a better solution. TS was proposed in its present form a few years ago. It has now become an established optimization approach that is rapidly spreading to many new fields. Together with other heuristic search algorithms, such as GA, TS has been singled out as “extremely promising” for the future treatment of practical applications. F. Glover has provided a complete user’s guide to tabu search. Ramon A. Gallego et al have used TS for network synthesis and they listed the issues in the tabu search. A.H. Mantawy et al have integrated tabu search with simulated annealing for the unit commitment problem.
Generally, the advantages of TS over other traditional optimization techniques can be summarized as follows:

(i) TS is characterized by its ability to avoid entrapment in local optimal solution and prevent cycling by using flexible memory of search history.

(ii) TS uses payoff (performance index or objective function) information to guide the search in the problem space. Therefore, it can easily deal with non-smooth, non-continuous, and non-differentiable objective functions that are the real-life optimization problems. Additionally, these properties relieve TS of assumptions and approximations, which often are required by traditional optimization methods for many practical optimization problems.

(iii) TS uses probabilistic transition rules to make decisions, not deterministic rules. Hence, TS is a kind of stochastic optimization algorithm that can search a complicated and uncertain area to find the global optimum. This makes TS more flexible and robust than conventional methods.

Typically, the TS algorithm starts with no knowledge of the correct solution, depending entirely on response from interacting environment to arrive at optimal solution.

2.6.2 Elements of Tabu Search

The basic elements of TS are briefly stated and defined as follows:

(i) **Current solution** \( (x_{current}) \): It is a set of the optimized parameter values at any iteration. It plays a central role in generating the neighbour trial solutions.

(ii) **Moves**: They characterize the process of generating trial solutions that are related to \( x_{current} \).
(iii) **Set of candidate moves, N (x_{current}):** It is the set of all possible moves or trial solutions, x_{trial}s in the neighborhood of x_{current}. In case of continuous variable optimization problems, this set is too large or even infinite set. Therefore, one could operate with a subset, S (x_{current}), with a limited number of trial solutions, nt, of this set.

(iv) **Tabu restrictions:** These are certain conditions imposed on moves that make some of them forbidden. These forbidden moves are listed to a certain size and known as tabu. This list is called the tabu list. The reason behind classifying a certain move as forbidden is basically to prevent cycling and avoid returning to the local optimum just visited. The tabu list size plays a great role in the search of quality solutions. The way to identify a good tabu list size is to simply watch for the occurrence of cycling when the size is too small and the deterioration in solution quality when the size is too large, caused by forbidden too many moves.

(v) **Aspiration Criterion (Level):** It is a rule that overrides tabu restrictions (i.e., if a certain move is forbidden by tabu restriction, the aspiration criterion, when satisfied, can make this move allowable). Different forms of aspiration criteria are used in the literature. The one considered here is to override the tabu status of a move if this move yields a solution, which has better objective function, than the one obtained earlier with the same move. The importance of using aspiration criterion is to add some flexibility in the TS by directing it toward the attractive moves.

2.7 **PARTICLE SWARM OPTIMIZATION (PSO)**

Particle swarm optimization is a stochastic, population-based search and optimization algorithm for problem solving. It is a kind of swarm intelligence that is based on social- psychological principles and provides insights into social behaviour, as well as contributing to engineering applications. The particle swarm optimization algorithm was first described in 1995 by James Kennedy and Russell
C. Eberhart. The techniques have evolved greatly since then, and the original version of the algorithm is barely used at present. Social influence and social learning enable a person to maintain cognitive consistency. People solve problems by talking with other people about them, and as they interact their beliefs, attitudes, and behaviour changes, the changes could typically be depicted as the individuals moving toward one another in a socio-cognitive space.

The particle swarm simulates a kind of social optimization. A problem is given, and some way to evaluate a proposed solution to it exists in the form of a fitness function. A communication structure or social network is also defined, assigning neighbours for each individual to interact with a population of individuals defined as random guesses as the problem solutions is initialized. These individuals are candidate solutions and are also known as the particles, hence the name particle swarm. An iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate solutions and remember the location where they had their best success. The individual's best solution is called the particle best or the local best. Each particle makes this information available to their neighbours. They are also able to see where their neighbours have had success. Movements through the search space are guided by these successes, with the population usually converging, by the end of a trial, on a problem solution better than that of non-swarm approach using the same methods.

The particle swarm optimization (PSO) algorithm is a population-based search algorithm inspired by the social behaviour of birds within a flock. The initial intent of the particle swarm concept was to graphically simulate the graceful and unpredictable choreography of a bird flock, the aim of discovering patterns that govern the ability of birds to fly synchronously, and to suddenly change direction with a regrouping in an optimal formation. From this initial objective, the concept evolved into a simple and efficient optimization algorithm. In PSO, individuals, referred to as particles, are "flown" through hyper dimensional search space. Changes to the position of particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals. The changes to a particle within the swarm are therefore influenced by the
experience, or knowledge, of its neighbours. The search behaviour of a particle is thus affected by that of other particles within the swarm therefore PSO is the kind of symbiotic cooperative algorithm. The consequence of modeling this social behaviour is that the search process is such that particles stochastically return toward previously successful regions in the search space. The operation of the PSO is based on the neighbourhood principle as social network structure.

2.8 EVOLUTIONARY PROGRAMMING (EP)

2.8.1 Overview

Evolutionary programming, originally conceived by Lawrence J. Fogel in 1960, is a stochastic optimization strategy similar to Genetic algorithms, but instead places emphasize on the behavioral linkage between parents and their offspring, rather than seeking to emulate specific genetic operators as observed in nature. Evolutionary programming is similar to Evolution strategies, although the two approaches developed independently. Like both ES and GAs, EP is a useful method of optimization when other techniques such as gradient descent or direct, analytical discovery are not possible. Combinatorial and real-valued function optimization in which the optimization surface or fitness landscape is "rugged", processing many locally optimal solution, are well suited for evolutionary programming. D.B. Fogel has given a detailed explanation about the Evolutionary programming and demonstrated it with suitable illustrations. H.T Yang et al have used the EP method for solving economic load dispatch with non-smooth fuel cost functions. L.L. Lai et al have demonstrated the effectiveness of EP by applying it for the reactive power planning with network contingencies. K.A. Juste et al have solved the unit commitment problem by the EP method.

In Evolutionary programming a strong behavioral link is sought between each parent and its offspring, at the level of the species. The general scheme of the EP follows the sequence below:

(i) An initial population of parents is generated at random.
(ii) Each parent is assigned a scaled objective function value or fitness score.

(iii) An offspring is created by altering each parent's elements with respect to a Gaussian distribution.

(iv) Each offspring is assigned a fitness score according to step (ii).

(v) For the entire population, parents and offspring included, a competition rule is defined.

(vi) Selection is then applied to determine which solutions have to be maintained into future generations.

(vii) The process proceeds to step (iii) (i.e. repeat generation, evaluation, competition and selection) unless the available execution time is exhausted or an acceptable solution has been discovered.

It should be pointed out that EP typically does not use any crossover as a genetic operator.

2.8.2 EP and GAs

There are two important ways in which EP differs from GAs.

(i) There is no constraint on the representation in EP. The typical GA approach involves encoding the problem solutions as string of representative tokens, the genome. In EP, the representation follows from the problem. A neural network can be represented in the same manner as it is implemented because the mutation operation does not demand a linear encoding.

(ii) The mutation operation simply changes the aspects of solution according to a statistical distribution which weights minor variations in the behavior of the offspring as highly probable and substantial. Further, the severity of mutations is often reduced as the global optimum is approached. There is a certain tautology here: if the
global optimum is not already known, how can the spread of the mutation operation be damped as the solutions approach it? Several techniques have been proposed and implemented which address this difficulty, the most widely studied being the “Meta-evolutionary” technique in which the variance of the mutation distribution is subject to mutation by a fixed variance mutation operator and evolves along with the solution.

2.8.3 EP and ES

Despite the independent development of Evolutionary programming and Evolution strategy over thirty years they share many similarities. When implemented to solve real valued function optimization problems both typically operate on the real values themselves (rather than any coding of the real values as is often done in GAs). Multivariate zero mean Gaussian mutations are applied to each parent in a population and a selection mechanism is applied to determine which solutions are to be removed from the population. Most of the theoretical results on convergence developed for ES or EP also apply directly to the other.

The main differences between ES and EP are:

(i) Selection: EP typically uses stochastic selection via a tournament. Each trial solution in the population faces competition against a preselected number of opponents and receives a “Win” if it is at least as good as its opponent in each encounter. Selection then eliminates those solutions with the least wins. In contrast, ES typically uses deterministic selection in which the worst solutions are purged from the population based directly on their function evaluation.

(ii) Recombination: EP is on abstraction of evolution at the level of reproductive populations (i.e., species) and thus no recombination mechanisms are typically used because recombination does not occur between species. In contrast, ES is on abstraction of evolution at the level of individual behaviour. When self-adaptive information is
incorporated this is purely genetic information (as opposed to phenotypic) and thus some forms of recombination are reasonable and many forms of recombination have been implemented within ES.

2.9 CONCLUSION

In this chapter the complete literature survey about the Genetic algorithms, Evolution strategy, Evolutionary programming, Genetic programming, Simulated annealing Tabu search and Particle Swarm Optimization and their application to the power system problems like unit commitment, economic load dispatch with emission constraints and maintenance scheduling is done. The advantages and issues related to each method were studied. From the above survey, it is inferred that the Evolutionary computing techniques are well suited for the solution of combinatorial optimization problems like unit commitment, economic dispatch and maintenance scheduling.