APPENDIX

1. PARTICLE SWARM OPTIMIZATION

Dr. Eberhart and Dr. Kennedy developed Particle Swarm Optimization (PSO) in 1995. PSO is a population based stochastic optimization technique. This computational technique is inspired by social behaviour of bird flocking or fish schooling. Inspired by biological system a lot of methods were already reported. For example, Artificial Neural Network (ANN) is a simplified model of human brain and Genetic Algorithm (GA) is inspired by the human evolution. PSO is a simulations of social system that denotes the collective behaviours of simple individuals interacting with their environment and each other. It may be known as Swarm Intelligence. Two popular swarm inspired techniques are Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) in computational intelligence area. ACO was inspired by the behaviours of ants and has many successful applications in discrete optimization problems. PSO, however, has found more applications than the ACO, for power system problems

PSO is an evolutionary computation technique like Genetic Algorithm (GA). In this technique, a group of random particles (solutions) are generated. According to fitness value the best solution is determined in the current iteration and also the best fitness values is stored. The best solution is known as pbest. Another best fitness value is also tracked in the iterations obtained so far. The best fitness value is a global best and its corresponding particle (solution) is called gbest. In every iteration all the particles are updated by the following pbest and gbest. Each particle updates its velocity and positions according to the following equations**:**

$$v_i^{k+1} = w * v_i^k + c1 * rand() * (pbest_i^k - x_i^k) + c2 * rand() * (gbest_i^k - x_i^k) \qquad \text{(AP1.1)}$$

Where,

x_i^k : Current position of ith particle at kth generation

v_i^k : Current velocity of ith particle at kth generation

$pbest_i^k$: pbest of ith particle for kth generation

$gbest_i^k$: gbest of ith particle considering the whole generation i.e. upto the kth generation

v_i^{k+1} : updated velocity of ith particle

w : inertia weight for ith particle

c1 & c2: constriction factors

$rand()$: random number between 0 and 1.

Using the above equation, a certain velocity that gradually gets close to pbest and gbest can be determined. There should be a limit of the particle velocity. The maximum value of the particle velocities is usually denoted as V_{max}. V_{max} is specified by the user according to the problem to be optimized. The calculated particle velocity is checked at each step to see whether it is exceeding the maximum velocity or not. If any particle velocity exceeds the maximum velocity limit, then the velocity is set to V_{max}.

The current searching point can be updated by the following equation:

$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad \text{(AP 1.2)}$$

The algorithm of the procedure as follows:

1. Initialize the particles
2. Calculate the fitness value of the particles
3. The best fitness value is selected and its corresponding particle is taken as pbest
4. Calculate particle velocity according to equation AP1.1
5. Limit the particle velocity that exceed the maximum velocity (V_{max})
6. Update Particle position according to equation AP 1.2
7. Choose the particle with the best fitness value of all the particles as gbest
8. Check stop condition i.e. either maximum iteration or minimum error criteria. If stop condition is satisfied, then stop else go back to step 2.

One of the great advantages of PSO over GA is that it takes real numbers as particles. Numbers of PSO parameters to be controlled are not many. List of parameters and their typical values are given below. Generally large numbers of particles are not required to get good results.

Constriction factors take very important role in PSO system. If c1 & c2 are not selected properly according to problem, the PSO system might not converge at all. For different problem different values or ranges are selected. Usually c1 equals to c2 and ranges from [0,4]. In few problems c1 and c2 equal to 2. in most cases c1 & c2 equal to 1.49445.

Inertia weight is another important factor for swarm optimization problems. Usually inertia weight is taken in the range of [0,1]. Inertia weight (w) must not be constant for better results. It is randomly selected within a certain range. Random selection of w provides successful tracking for a dynamic optimization problem. To determine w in [25] introduce a formula given below.

$$w = w_{\max} - \frac{(w_{\max} - w_{\min}) * present\_iteration}{\max imum\_iteration} \qquad \text{(AP 1.3)}$$

Where,

$w_{\max} = 0.9$ and $w_{\min}$ =0.4

Here *w* varies linearly from 0.9 to 0.4 with the increase of iteration. It has been observed that inertia weight depends on the selection of maximum error. When tracking a dynamic system it cannot be predicted whether a larger inertia weight or a smaller inertia weight will be better at any given time. So *w* should not depend on maximum iteration and must not be fixed. It may vary roughly within 0.5 and 1. But from clerc's constriction method, *w* becomes constant at 0.729. So slight adjustment is made to inertia weight.

$$w = 0.5 + rand()/2 \qquad \text{(AP 1.4)}$$

Equation (AP 1.4) produces a number randomly varying between 0.5 and 1.0, with a mean of 0.75.

The stop condition may be either the maximum number of iteration to be executed or the required minimum error. Generally in most of the problems the required minimum error is set.

PSO is one-way information sharing mechanism. Here gbest/pbest gives the information to the others. In PSO, individuals who fly past optima are tugged to return toward them, good solutions are retained by all the particles. PSO only looks for the best solution. All the particles have a tendency to converge to the best solution quickly in most cases.

2. GENETIC ALGORITHM

Genetic Algorithms are inspires by Darwin's theory of evolution. Solution to a problem solved by genetic algorithms uses an evolutionary process. Algorithm begins with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. The crossover and mutation are the most important mechanism of the genetic algorithm.

The vector, representing the problem variables, is referred to as chronosome in GA literature. Encoding of a chromosome is also an important factor. The most used way of encoding is a binary string. A chromosome then could look like this:

Chromosome 1	1001100100110110
Chromosome 2	1101111000011110

Of course, there are many other ways of encoding. The encoding depends mainly on the solved problem. For example, one can encode directly integer or real numbers, sometimes it is useful to encode using a continuation as well.

After encoding of the chromosomes, crossover operation is applied for the reproduction of new solutions (offsprings). Reproduction is a process in which individual chromosomes are copied according to their objective function values. Biologists call this function as fitness function. Copying chromosomes according to their fitness values means that chromosomes with a higher value have a higher probability of contributing one or more offspring in the next generation. This operation of course is an artificial version of natural selection. There are many methods in selecting the best chromosomes. After the selection of two parent chromosomes, some crossover point is randomly chosen and new solutions are generated by taking a copy of everything before this point from the first parent and then copy everything after the crossover point from the other parent. Crossover can be illustrated as follows: (| is the crossover point):

Chromosome 1	10011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	10011 | 11000011110
Offspring 2	11011 | 00100110110

There are other ways to implement crossover, for example we can choose more crossover points. Crossover can be quite complicated and depends mainly on the encoding of chromosomes. Specific crossover made for a specific problem can improve performance of the genetic algorithm.

The mutation operator plays also an important role in the simple GA. Mutation rates is generally smaller. After a crossover is performed, mutation takes place. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the solved problem. Mutation operation randomly changes the offspring resulted from crossover. In case of binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. Mutation can be then illustrated as follows:

Original offspring 1	1001111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1000111000011110
Mutated offspring 2	1101101100110110

The technique of mutation (as well as crossover) depends mainly on the encoding of chromosomes. For example when we are encoding permutations, mutation could be performed as an exchange of two genes.

The outline of the basic Genetic Algorithm as follows:

1. Generate population of the chromosomes randomly.
2. Evaluate objective function (fitness value) of each chromosome in the population.
3. Select two parent chromosomes from a population according to their fitness.
4. Cross over the parents with a crossover probability to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
5. Mutate new offspring with a mutation probability at each locus (position in chromosome).
6. Replace new offspring in the new population.
7. If end condition is satisfied, then stop and show the results, otherwise go to step 2.

Two basic parameters of GA are crossover probability and mutation probability. There are some other parameters of GA like population size, selection of chromosomes as parents, Encoding of chromosomes. List of parameters and their typical value is given below:

Crossover probability indicates how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survive to next generation. Crossover rate should be high generally, about 80%-95%. However some results show that for some problems crossover rate about 60% is the best.

Mutation probability stands for how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover (or directly copied) without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search. Mutation rate should be very low. Best rates seem to be about 0.5%-1%.

Population size refers to how many chromosomes are in population (in one generation). If there are too few chromosomes, GAs have few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit (which depends mainly on encoding and the problem) it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations. Good population size is about 20-30, however sometimes sizes 50-100 are reported as the best. Some research also shows, that the best population size depends on the size of encoded string (chromosomes). It means that if you have chromosomes with 32 bits, the population should be higher than for chromosomes with 16 bits.

Chromosomes are selected from the population to be parents for crossover. The problem is how to select these chromosomes. According to Darwin's theory of evolution the best ones survive to create new offspring. There are many methods in selecting the best chromosomes. Examples are roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.

For efficient and effective search meta-heuristic techniques are widely applied in business, scientific and engineering circles. This algorithm is computationally simple but powerful in search space for improvement. The advantages of GA are that it is not restricted by unimodality, derivatives or continuity.

Calculus-based methods depend upon the existence of derivatives. These methods are local in scope and the optima they seek are the best in a neighborhood of the current point. Even if we allow numerical approximation of derivatives, this is a severe shortcoming. For searching researchers avoid discontinuities, vast multimodal and noisy search space. Methods depending upon the restrictive requirements of continuity and derivative existence are unsuitable. So calculus based methods are rejected for its restricted requirements and because of its inherently local scope of search. It is insufficiently robust in unintended domains. Due to shortcomings of calculus-based schemes, random search algorithms are becoming popular. GA and PSO are examples of a random search method that uses random choice as a tool to guide a highly exploitative search through a coding of a parameter space.