Report of

Mini-Project on Optimization of Economic Dispatch Problem using GA

Performed under Virtual Internship Programme, RIT

Under the guidance of -

Dr. H.T. Jadhav

Prof. P.D. Bamane

Submitted by – Mr. Maneesh Vidyadhar Sutar

Date of Submission – 27th July 2020

Abstract

Today, we know that the world's energy demand has risen sharply and engineers around the world are finding ways to increase the generation while decreasing its cost. Solving the economic dispatch problem is the first step in it so that the generators can fulfill the desired energy demand, with minimum cost.

We are given a problem that includes either all thermal generators or thermal generators with wind farm integration, and we have used advanced optimization techniques to find the optimal working point of all individual generators (that is the value of power generated individually), with minimum constraint violation.

After applying the genetic algorithm for optimization, we found out the minimum cost of generation from thermal generators and wind integrated thermal generators, while maintaining the desired power supply demand. The violation of constraint was nearly equal to zero. Also, the cost of generation was much less in the case of a wind farm integrated system.

Table of Contents

Introduction

As our electricity demand increases day by day, it is necessary to find the best possible ways to generate enough power to reach that power demand. But increasing the generation means that cost of generation will also increase. Also, not all generators work at 100% efficiency, or not all the generators work at even the same efficiency. All power plants vary in their peak generation capacity. Also, when load demand falls suddenly, engineers should decide which generator may lower their generation, while considering the cost of restarting the generators to match peak demand. The sizes of the electric power system are increasing rapidly to meet the energy requirement. So, the number of power plants are connected in parallel to supply the system load by an interconnection of the power system. In the grid system, it becomes necessary to operate the plant units more economically.

Many factors play a role to determine the most cost-efficient way to generate electricity, and economic load dispatch is the study of the same. The Economic Load Dispatch means the real and reactive power of the generator varies within certain limits and fulfills the load demand with less fuel cost. In solving the ED problem, one seeks to find the optimal allocation of the electrical power output from various available generators. To find that optimal point of operation, we will use the Genetic Algorithm, which is a heuristic search and optimization technique.

- For this project, we are provided with mathematical expressions of the cost of generation of each thermal power plant and power limits within which each power plant operates. Total power demand was also provided.
- At first, we will calculate the cost of operation using the gradient method [1, pp 267-270] to get an idea of what result should look like.
- Then, we will implement the Genetic Algorithm to find the minimum cost of operating thermal power plants, by varying the power generation of individual plants.
- Then, the last thermal power plant will be replaced by a wind farm and its properties are given.
- So, for implementation of the wind farm, we will refer the paper by “An Economic Dispatch Model Incorporating Wind Power” by John Hetzer [2] (which is referred as “Hetzer Model” from here on)

- We will apply the Hetzer Model to calculate the probable cost of generation from the wind farm, combine it with the cost of generation of other thermal power plants, and apply the Genetic Algorithm to minimize the overall cost.

The Problem Statement

In the given problem, there are 3 thermal generators, whose input-output characteristics are as follows:

$$f_1 = 0.004\, P_{G1}^2 + 5.3\, P_{G1} + 500 \text{ Btu/h}$$
$$f_2 = 0.006\, P_{G2}^2 + 5.5\, P_{G2} + 400 \text{ Btu/h}$$
$$f_3 = 0.009\, P_{G3}^2 + 5.8\, P_{G3} + 200 \text{ Btu/h}$$

Where, P_{G1}, P_{G2} and P_{G3} represent the power generated by respective thermal power plants. We must use the Genetic Algorithm and vary these parameters to minimize the cost.

The total power demand and generation limits act as constraints. The total power demand is 800 MW. The power limits of generators are given as:

$$200 \leq P_{G1} \leq 450$$
$$150 \leq P_{G2} \leq 350$$
$$100 \leq P_{G3} \leq 225$$

We need to find the solution for 2 cases:

I. Considering all generators as thermal generators.

II. Replacing the last generator by wind power generator with the following properties: Number of turbines = 50, Power rating of each turbine is 4 MW. Cut in velocity (Vin) = 3; rated velocity (Vr) = 10; Cut-out velocity (Vout) = 20; wind direct cost coefficient = 0.08; penalty cost coefficient (Cpwj) = 10; reserved cost coefficient (Crwj) = 4; shape factor = 2; and scale factor = 9.

Theoretical Background

Let's look at theoretical concepts behind GA and Hetzer Model.

Genetic Algorithm:

Genetic Algorithm is the heuristic search and optimization technique that mimics the process of natural evolution. It is a subset of Evolutionary Computation. It is mostly used to find optimal solutions to problems for which derivative info is not available. GA was developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg.

In GAs, we start with a population, which is nothing but a set of a possible solution to the given problem. Each solution is known as a chromosome. These chromosomes are calculated for their fitness value. Then we perform crossover and mutation operation on them, and this process goes on for generations. The chromosomes with more fitness value survive each generation. The termination criteria for this algorithm might be:

1. Fixed number of generations reached
2. Desired fitness value achieved

Advantages of Genetic algorithm

1. No need for derivative information.
2. It can be used to optimize both continuous and discrete functions.
3. It provides a list of "good" solutions and not just a single solution.
4. It's useful when the search space is exceptionally large and there are many parameters involved.

Limitations of GAs

1. When problems are simple and derivative information is available, then GA is not a good option.
2. Fitness value calculation after each generation and a considerable number of populations may require large computational sources and memory.
3. There is no guarantee of an optimal solution. Sometimes GA may not give an optimal solution if not implemented properly.

Basic Structure of GA:

1. Initialize the population randomly.
2. Calculate the cost value.
3. While termination criteria not reached:
    a. Calculate the fitness value.
    b. Select parents out of the population, perform crossover and create offspring.
    c. Select parents, perform mutation to create new offspring
    d. Survival selection based on the fitness value or age of all the population.
4. Return the best parameters and the best cost.

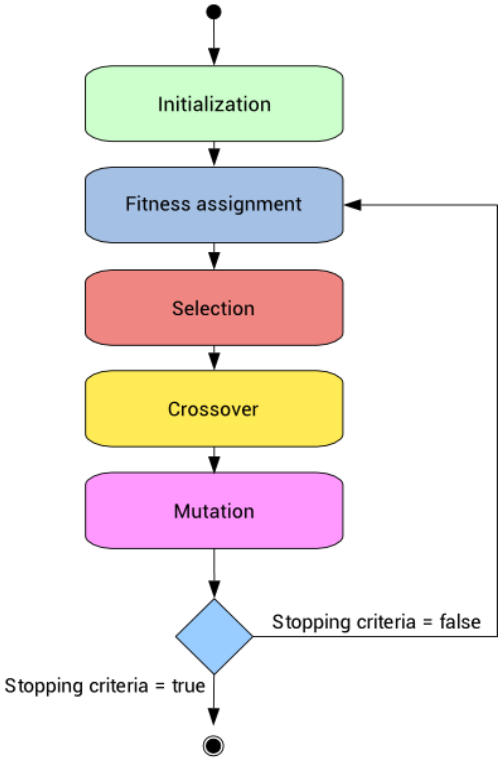

Fig. 1 – Structure of Genetic Algorithm

Hetzer Model:

In the case of a Thermal Generator, we can control the amount of coal/gas and thus can control the input to the generator. But for Wind Turbines, nobody can predict the wind flow which will drive the generator; thus, we can't control turbine input. Wind velocity is never constant. Also, the wind turbine has its limits in which it can generate electricity. Those are:

Cut-in speed (Vin): Minimum wind velocity required to generate electricity

Rated speed (Vr): The mi mum velocity at which turbine produces rated power

Cut-out speed (Vout): The maximum velocity of the turbine before cutting off.

In the ED problem, we must set the value of the Scheduled Power of wind farm, such that the overall cost of operation (thermal + wind) as a minimum. Scheduled Power is always between 0 and rated power. Now, if the power generated by a wind turbine is more than scheduled power (due to higher wind speeds), it is known as *Underestimation*. In this case, thermal generators (running in parallel) must slow down or stop their generation. If the power generated by a wind turbine is less than scheduled power (due to lower wind speeds), it is known as *Overestimation*. In such cases, thermal generators must increase their generation to reach the desired power demand.

Thus, there's a cost present in both overestimation and underestimation of scheduled power. We must consider both the costs. In the Hetzer Model [2], Weibull Probability Distribution is considered to find the probable wind speed, for both overestimation and underestimation. Then, we integrate over minimum and maximum power limits to find the underestimation and overestimation costs separately. Then, we add direct wind cost, underestimation, and overestimation cost, which becomes the total cost of wind generation.

MATLAB Implementation

Case I - 1: All Thermal Power Plants with Gradient Method

```
clear all;
clc;
close all;
lambda = 2.0;          % initial lambda value (assumed)
PD = 800;              % total dispatched power
BB = [ 0 ; 0 ; 0 ];       % no transmission loss given
alpha = [ 500 400 200 ];
beta = [ 5.3 5.5 5.8 ];
```

```
gamma = [ 0.004 0.006 0.009 ];
X_limit=[200 450
        150 350
        100 225];
D = length (X_limit(:,1));
P = zeros(1,D);
X_min = X_limit(:,1)';
X_max = X_limit(:,2)';
del_P = 0;
del_lambda = 0;
PL = 0;
max_iter = 100;
W = zeros(max_iter,1);

for k = 1:max_iter
   for i = 1:D
      P(i) = ( lambda - beta(i) ) / ( 2 * ( gamma(i) + lambda *  BB(i) ) ) ;
      if P(i) < X_min(i)
         P(i) = X_min(i);
      elseif P(i) > X_max(i)
         P(i) = X_max(i);
      end
   end

   del_P = PD + PL - sum(P);

   del_lambda = del_P / sum(1./(2*gamma));
   lambda = lambda + del_lambda;

   W(k) = del_P;

   fprintf('iter = %i   P_generated = %f   P1 = %f   P2 = %f  P3 = %f\n', k, sum(P), P(1) , P(2) , P(3) );
end

final_cost = sum(alpha +  beta.*P +gamma.* (P.^(2)) );

figure;
plot(W,'LineWidth',2);
xlabel('Iteration');
```

```
ylabel('del_P');
grid on;
```

Case I - 2: All thermal Power Plants with Genetic Algorithm

Main Body:

```
clc;
clear;
close all;
```

```
% Problem Statement Parameters
X_limit=[100 200
        100 400
        100 400];
D = length (X_limit(:,1));

X_min = X_limit(:,1)';
X_max = X_limit(:,2)';

PD = 800;
alpha = [8 7 5];
beta = [0.8 0.7 0.6];
gama = [0.008 0.007 0.006];
lambda = 1000;          % voilation cost multiplier
```

```
% GA parameters
Max_iter = 1000;
np = 400;
pc = 0.7;
mu = 0.2;
nc = round(pc*np/2);              % number of crossovers
nm = round(np*mu);
sigma=0.1;
```

```
% Initialization
X = repmat (X_min, np ,1) + rand(np,D).* (repmat (X_max, np ,1) - repmat (X_min, np ,1));

Z = cost(X,alpha, beta, gama, PD , lambda);
```

```
[best_cost,ii] = min(Z);
best_parameter= X(ii,:);
```

```
% Main loop
prob = zeros(np,1);
W = zeros(1,Max_iter);

for it = 1:Max_iter
    % selection probabilities
    fitness_function = 1./(1 + Z);
    for k = 1:np
        prob(k,:) = fitness_function(k,:)/sum(fitness_function);
    end

    Zcross=zeros(nc,2);
    Y1 = zeros(nc,D);
    Y2 = zeros(nc,D);



    for j = 1:nc
        % select parents
        P1 = RouletteWheelSelection(prob);
        P2 = RouletteWheelSelection(prob);

        % perform crossover
        [Y1(j,:),Y2(j,:)] = uniform_crossover (P1, P2, D, X , X_min, X_max);          % create 2 offspring
        Zcross(j,1) = cost(Y1(j,:),alpha, beta, gama, PD , lambda);        % sphere value of 1st offspring
        Zcross(j,2) = cost(Y2(j,:),alpha, beta, gama, PD , lambda);        % sphere value of 2nd offspring
    end
    Ycross=[Y1;Y2];            % convert all to a column matrix [nc*2,2]
    Zcross=Zcross(:);          % convert all to a column matrix [nc*2,1]

    y=zeros(nm,D);
    Zmutate=zeros(nm,1);
    for i=1:nm
        % Select Parent
        m=randi([1 np]);
        p=X(m,:);

        % Apply Mutation
```

```
        y(i,:) = mutate (p, D, sigma);

        Zmutate(i,:) = cost(y(i,:),alpha, beta, gama, PD , lambda);
    end
    Ymutate=y;
    % Create Merged Population
    Zga=[Z
        Zcross
        Zmutate];          % shape [np + nc*2 + nm , 1]
    Yga=[X
        Ycross
        Ymutate];               % shape [np + nc*2 + nm , 3]

    % Sort Population
    [Zga, SortOrder]=sort(Zga);
    Yga=Yga(SortOrder,:);

    % Update Worst Cost
    WorstCost=max(Zga);

    % Truncation
    Z=Zga(1:np,:);
    Y=Yga(1:np,:);

    % Store Best Solution Ever Found
    best_iter_parameter=Yga(1,:);

    % Store Best Cost Ever Found
    best_iter_cost_value = Zga(1,:);

    if best_iter_cost_value < best_cost
        best_cost = best_iter_cost_value;
        best_parameter = best_iter_parameter;
    end
    W(it) = best_cost;
    % Show Iteration Information
    fprintf('it = %f best_cost_value = %f X1 = %f X2 = %f X3 = %g\n', it, best_cost, best_parameter)

end
```

```
% Results

figure;
plot(W,'LineWidth',2);
xlabel('Iteration');
ylabel('Cost');
grid on;

final_cost=cost(best_parameter,alpha, beta, gama,PD , lambda);
voilation = constraint(best_parameter , PD);
generation_cost = final_cost - lambda*voilation;
```

Cost Function:

```
function Z = cost(X1,alpha, beta, gama, PD , lambda)
Voilation = constraint(X1,PD);
yy = length (X1(:,1));
generation_cost = zeros(yy,1);
for co = 1:yy
   generation_cost(co) = generation_cost(co) + sum(alpha + beta.*X1(co,:)+gama.*(X1(co,:).^(2)));
end
Z = generation_cost + lambda .* Voilation;
end
```

Constraints:

```
function Voilation = constraint(X,PD)
Pi = sum(X,2);
Voilation = zeros(length(Pi),1);
for cv = 1: length(Pi)
   if (Pi(cv)~=PD)
      Voilation(cv)=abs((Pi(cv)-PD));
   end
end
```

Roulette Wheel Selection:

```
function ij = RouletteWheelSelection(prob)
   r=rand;

   c=cumsum(prob);

   ij=find(r<=c,1,'first');
end
```

Uniform Crossover:

```
function [Y1, Y2] = uniform_crossover (P1, P2,D, X, X_min, X_max)
zeta = rand (1,D);
Y1 = zeta.*X(P1,:) + (1-zeta).*X(P2,:);
Y2 = zeta.*X(P2,:) + (1-zeta).*X(P1,:);

% check boundary condition
a=find (Y1<X_min);
Y1(a) = X_min(a);
b=find (Y1>X_max);
Y1(b) = X_max(b);
c=find (Y2<X_min);
Y1(c) = X_min(c);
d=find (Y2>X_max);
Y1(d) = X_max(d);
end
```

Mutation:

```
function y = mutate (p, D, sigma)
j = randi(D);
y = p;
r = randn(size(p));
y(j) = p(j)+ r(j)*sigma;
end
```

Case II: 2 Thermal Power Plants and a Wind Farm.

Main Body:

```
clc;
clear;
close all;
```

```
% Thermal Power Plants Parameters
X_limit=[200 450
        150 350];
X_min = X_limit(:,1)';
X_max = X_limit(:,2)';
PD = 800;
alpha = [ 500 400 ];
beta = [ 5.3 5.5  ];
gama = [ 0.004 0.006 ];
```

```
% Wind Power Plants Parameters
No_Turbine = 50;
Pr = 4; % rated power each turbine
Vin = 3;
Vr = 10;
Vout = 20;
direct_wind_cost_coe = 0.08;
Cpwj = [10];    % penalty cost
Crwj = [4];     % reserved supply cost
shape_factor = 2;   % k
scale_factor = 9;    % c

rated_wind_power = No_Turbine * Pr;
ratio_of_velocity = (Vr - Vin)/Vin;

X_min(1,3) = 0;
X_max(1,3) = rated_wind_power;

D = length(X_min(1,:));
```

```
% GA Parameters
Max_iter = 1000;
np = 200;
```

```
pc = 0.7;
mu = 0.2;
nc = round(pc*np/2);            % number of crossovers
nm = round(np*mu);
sigma=0.1;
lambda = 1000;          % voilation cost multiplier
```

Codes for Initialization and Main Loop are the same as that of case I.

Cost function:

```
function Z = cost(X1,alpha, beta, gama, PD , lambda)
Voilation = constraint(X1,PD);
yy = length (X1(:,1));
thermal_generation_cost = zeros(yy,1);
for co = 1:yy
   thermal_generation_cost(co) = sum(alpha + beta.*X1(co,1:2)+gama.*(X1(co,1:2).^(2)));
end

wind_generation_cost = zeros(yy,1);

for co = 1:yy
   Scheduled_wind_power = X1(co,3);
   wind_generation_cost(co) = windcost(Scheduled_wind_power);
end
Z = thermal_generation_cost + wind_generation_cost + lambda .* Voilation;
end
```

Wind Cost function:

```
function wind_total_cost = windcost(Scheduled_wind_power)
   No_Turbine = 50;
   Pr = 4; % rated power each turbine
   Vin = 3;
   Vr = 10;
   Vout = 20;
   direct_wind_cost_coe = 0.08;
   Cpwj = [10];    % penalty cost
   Crwj = [4];     % reserved supply cost
   shape_factor = 2;   % k
```

```
    scale_factor = 9;    % c
    % Wind calculation. w = power. wr = rated power
    Prw0 = 1 - exp(-(Vin/scale_factor)^(shape_factor)) + exp(-(Vout/scale_factor)^(shape_factor));  % power at
w = 0
    Prwwr = exp(-(Vr/scale_factor)^(shape_factor)) - exp(-(Vout/scale_factor)^(shape_factor));      % power at w
= wr

    rated_wind_power = No_Turbine * Pr;
    ratio_of_velocity = (Vr - Vin)/Vin;

    Prwwu = @ (w)   (w -  Scheduled_wind_power)/rated_wind_power *
(shape_factor*ratio_of_velocity*Vin/scale_factor)*(((1+(w/rated_wind_power*ratio_of_velocity))*Vin/scale_f
actor)^(shape_factor-1))*(exp(-
(((1+(w/rated_wind_power*ratio_of_velocity))*Vin/scale_factor)^(shape_factor))));
    wundesti = integral(Prwwu, Scheduled_wind_power, rated_wind_power, 'ArrayValue', true);

    Wind_underestimation_cost = Cpwj * wundesti + (rated_wind_power-
Scheduled_wind_power)*Prwwr*Cpwj;

    Prwwo = @ (w)   (Scheduled_wind_power-w)/rated_wind_power*
(shape_factor*ratio_of_velocity*Vin/scale_factor)*(((1+(w/rated_wind_power*ratio_of_velocity))*Vin/scale_f
actor)^(shape_factor-1))*(exp(-
(((1+(w/rated_wind_power*ratio_of_velocity))*Vin/scale_factor)^(shape_factor))));
    woveresti = integral(Prwwo, 0, Scheduled_wind_power, 'ArrayValue', true);

    Wind_overestimation_cost = Crwj * woveresti + (Scheduled_wind_power-0)*Prw0*Crwj;

    direct_cost = Scheduled_wind_power*direct_wind_cost_coe;


    wind_total_cost = Wind_underestimation_cost + Wind_overestimation_cost + direct_cost;
end
```

Codes for Constraints, Roulette Wheel Selection, Crossover, and Mutate are similar to Case I.

Results and Analysis

Case I: All Thermal Power Plants.

In this case, I solved the problem using 2 methods, traditional Gradient Approach, and Genetic Algorithm.

While using the Gradient Method, I assumed lambda = 2.0 initially, then ran the code for 100 iterations. After which, I got the violation curve (delta P vs iteration) and cost curve (cost v/s iteration) as follow:

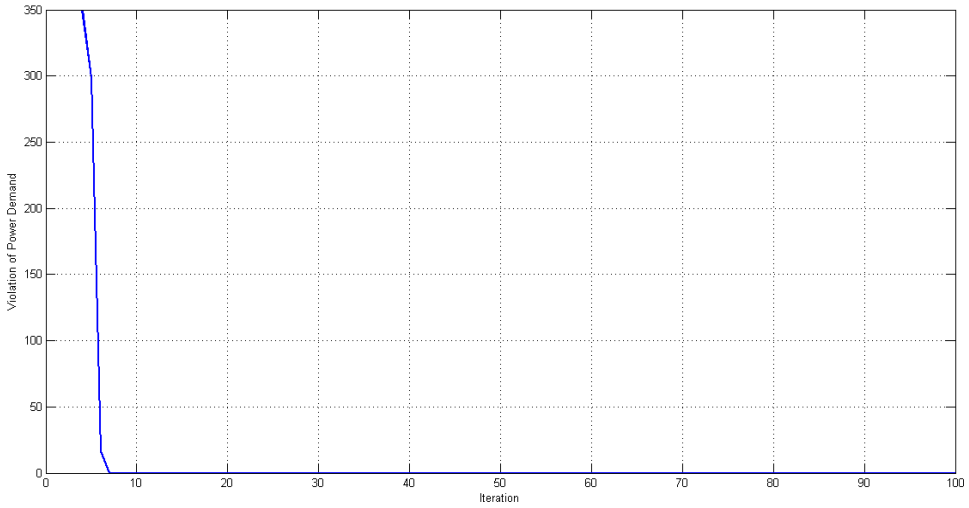

Fig. 2 – Violation of Constraint (del_P) v/s Iterations Curve for Gradient Method

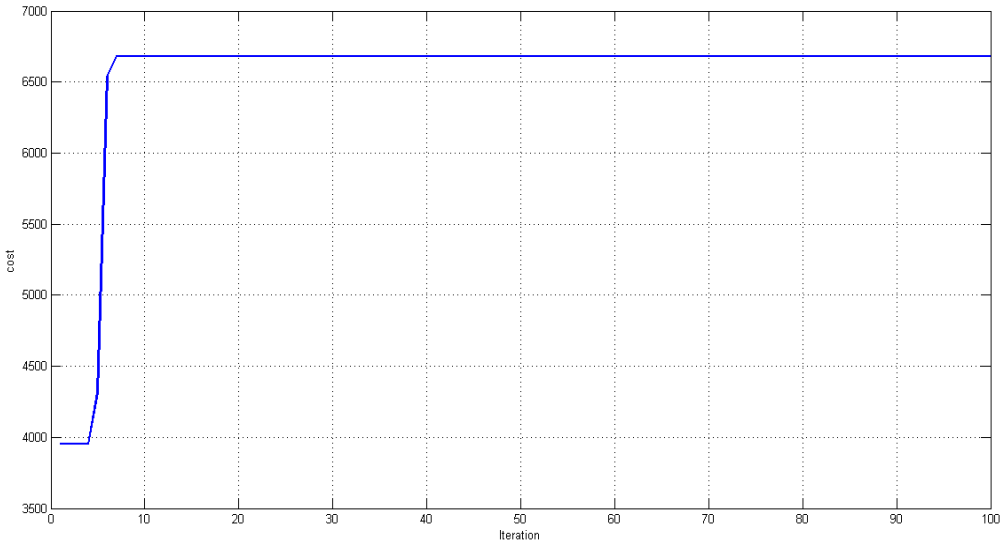

Fig. 3 – Total Cost of Generation v/s Iterations Curve for Gradient Method

Then, I applied the Genetic Algorithm to the same problem. I chose the number of iterations as 1000 and the number of population = 400. The lambda for constraint violation was set at 1000.

For that, I got convergence characteristic as follow:

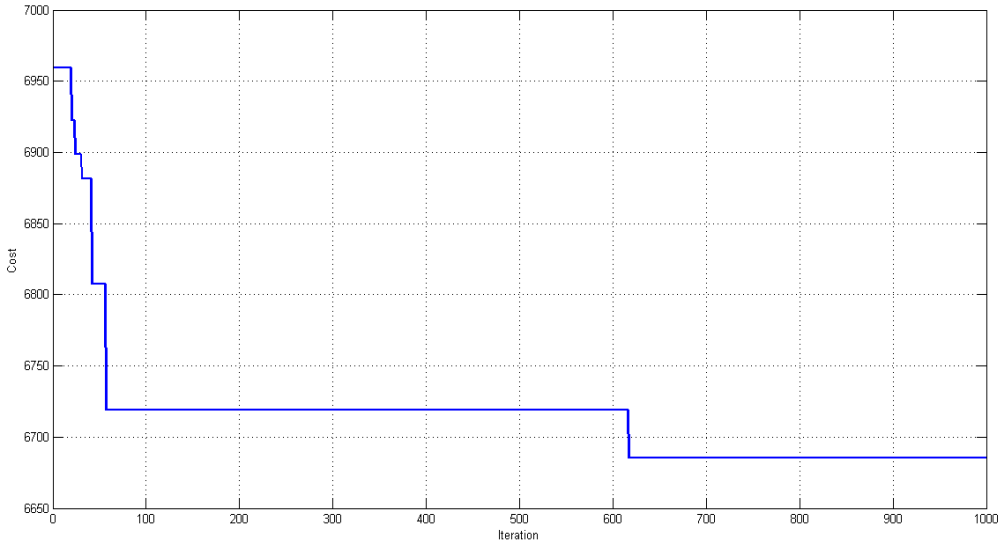

Fig. 4 – Cost of Generation v/s Iteration Curve for Case I GA

Case II: Two Thermal Power Plants and One Wind Turbine:

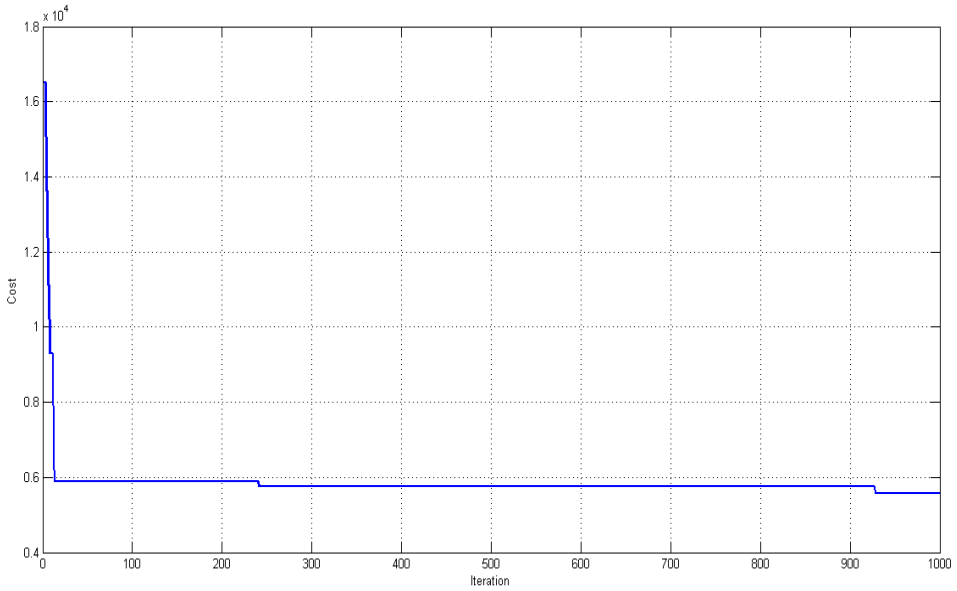

Fig. 5 – Cost of Generation v/s Iteration Curve for Case II

I replaced the last thermal generator in the previous problem with a wind farm with given specifications. In the cost function, I added the equations which calculate the power generation cost of a wind farm according to the Hetzer Model. Applying GA for 1000 iterations and 200 population, I got convergence characteristics as shown in fig. 5.

This is a comparison between all the cases:

Cases		Case I: All 3 are thermal power plants		Case II: 2 Thermal + 1 Wind Power Plant
Optimization Method		Gradient Method	Genetic Algorithm	Using Genetic Algorithm
Total Power Generated		800 MW	799.9998 MW	799.9949 MW
Violation of Constraint		0 MW	0.000154 MW	0.0051 MW
Individual Generation	P1	400 MW	387.9430 MW	351.7644 MW
	P2	250 MW	259.3393 MW	267.4429 MW
	P3	150 MW	152.7175 MW	180.7876 MW (wind generation)
Total cost of generation		6682.5	6685.2	5561.5
Cost of only thermal generation		6682.5	6685.2	5159.4
Cost of Wind Generation		0	0	351.2428

Table 1 – Comparison between Gradient Method and Genetic Algorithm in both cases

Analysis:

1. The gradient method gives the most accurate results with zero violation of constraint.
2. Since violation in the Genetic Algorithm tends to zero, we can be assured of getting the near-optimal solution to the given problem.
3. Looking at convergence characteristics of GA, we can see that the cost drops suddenly during the initial iterations, and then it's almost constant over the next hundreds of iterations.
4. The total cost of generation in case II is much less than that in Case I.

5. Because of wind farm integration in case II, thermal generation cost has also decreased sharply, which means less coal to burn.

Conclusion and Future Scope:

1. We have solved the Economic Dispatch Problem considering 3 thermal generators and a wind farm, using the Genetic Algorithm. Here, transmission line losses were not considered. So, we can include them and solve for the optimal solution.
2. Since GA is a heuristic search algorithm, the solution obtained may not be the best one. We compared the results of the GA method with the Gradient Method and found this to be true. To overcome this, we might choose a different combination of parameters such as maximum iterations, lambda, probability of crossover, etc.
3. We used the Hetzer Model to find the cost of wind generation at scheduled power. Here, the wind farm was considered compact (that is total rated power was simply the number of turbines multiplied by rated power of single turbine) So we can further work on the same problem while considering the individual properties of each turbine.
4. The results also show that cost of power generation when integrated with Wind Farm (case – II) is much less than the cost of generation with only thermal power plants (case-I). Thus, even though Wind Farm requires a huge amount of pre-investment, the ROI might be higher considering long term savings in a generation. Also, coal is a non-renewable source of energy and its complete exhaustion is near. Thus, it's a need of time to move on to renewable energy sources like Wind, Solar, etc. for power generation.

References

[1] Hadi Saadat, *Power System Analysis*. New York: McGraw Hill, 1999.

[2] John Hetzer, David C. Yu, Member, IEEE, and Kalu Bhattarai, Member, IEEE, "An Economic Dispatch Model Incorporating Wind Power," in *IEEE Transactions on Energy Conversion, Vol. 23, No. 2, June 2008*