POWER FLOW OPTIMIZATION USING SEEKER OPTIMIZATION ALGORITHM AND PSO

VIGNESH.P
Department of Electrical & Electronics Engineering
Anna University
Veerammal Engineering College, Dindigul, Tamilnadu
India
vickyme@gmail.com

Abstract: - The objective of an Optimal Power Flow (OPF) algorithm is to find steady state operation point which minimizes generation cost, loss etc. or maximizes social welfare, loadability etc. while maintaining an acceptable system performance in terms of limits on generators’ real and reactive powers, line flow limits, output of various compensating devices etc.

This paper aims to find a solution for the Optimal Power Flow (OPF) problem with security constraints by the seeker optimization algorithm (SOA). The major objective is to minimize the overall operating cost and real power losses while satisfying the power flow equations and system security. The SOA is based on the concept of simulating the act of human searching, where the search direction is based on the empirical gradient by evaluating the response to the position changes. To demonstrate its robustness, the proposed algorithm was tested on IEEE 30 Bus System and the proposed approach results were simulated using MATLAB. The performance of this algorithm is compared with the performance of PSO.

Key-Words: - global optimization, power system, seeker optimization algorithm

1 Introduction
The optimal power flow is an inherently non-linear optimization problem with a non-linear objective function and a set of non-linear equality and inequality constraints. Optimal power flow is one of the main functions of power generation operation and control. It determines the optimal setting of generating units[1]. It is therefore of great importance to solve this problem as quickly and accurately as possible.

In the past two decades, a wide variety of optimization techniques have been applied in solving the optimal power flow problem, such as linear programming, non-linear programming, Newton-based techniques, and interior point methods. The optimal power flow problem is a highly non-linear and a multi-modal optimization problem which exists more than one local optimum. Hence, most of these techniques are not suitable for such a problem. Recently, genetic algorithm and particle swarm optimization (PSO) have been proposed for solving the optimal power flow problem[3]. Particle swarm optimization, first introduced by Eberhart and Kennedy, has been used extensively in many fields, including function optimization, neural network and system control, etc. Unfortunately, the premature convergence of particle swarm optimization algorithm degrades its performance.

2 Problem formulation
Optimal Power flow (OPF) is allocating loads to plants for minimum cost while meeting the network constraints. It is formulated as an optimization problem of minimizing the total fuel cost of all committed plant while meeting the network (power flow) constraints. The variants of the problems are numerous which model the objective and the constraints in different ways.

The basic OPF problem can described mathematically as a minimization of problem of minimizing the total fuel cost of all committed plants subject to the constraints[1].

Minimize \( \sum_{i=1}^{n} F_i(P_i) \)  

(1)

\( F_i(P_i) \) is the fuel cost equation of the ‘i’th plant. It is the variation of fuel cost ($ or
Rs) with generated power (MW). Normally it is expressed as continuous quadratic equation
\[ F_i(P_i) = a_i P_i^2 + b_i P_i + c_i, \quad P_i^{\text{min}} \leq P_i \leq P_i^{\text{max}} \]  
(2)

The total generation should meet the total demand and transmission loss. The transmission loss can be determined from power flow.
\[ \sum_{i=1}^{n} P_i = D + P_L \]  
(3)

3 General reviews

In 1990, Chowdhury did a survey on economic dispatch methods. In 1999, J.A.Momoh et al. presented a review of some selected OPF techniques. In last decades, the following optimization methods has been used widely.

[1] Linear Programming (LP) method, Newton Raphson (NR) method, Quadratic Programming (QP) method, Nonlinear Programming (NLP) method, Interior Point (IP) method and Artificial Intelligence (AI) methods.

Even though, excellent advancements have been made in classical methods, they suffer with the following disadvantages: In most cases, mathematical formulations have to be simplified to get the solutions because of the extremely limited capability to solve real-world large-scale power system problems. They are weak in handling qualitative constraints. They have poor convergence, may get stuck at local optimum, they can find only a single optimized solution in a single simulation run, they become too slow if number of variables are large and they are computationally expensive for solution of a large system.

Whereas, the major advantage of the AI methods is that they are relatively versatile for handling various qualitative constraints. AI methods can find multiple optimal solutions in a single simulation run. So they are quite suitable in solving multi-objective optimization problems. In most cases, they can find the global optimum solution. The main advantages of ANN are: Possesses learning ability, fast, appropriate for non-linear modeling, etc. Where as, large dimensionality and the choice of training methodology are some disadvantages of ANN. The advantages of Fuzzy method are: Accurately represents the operational constraints and fuzzified constraints are softer than traditional constraints. The advantages of GA methods are: It only uses the values of the objective function and less likely to get trapped at a local optimum. Higher computational time is its disadvantage. The advantages of EP are adaptability to change, ability to generate good enough solutions and rapid convergence. ACO and PSO are the latest entry in the field of optimization. The main advantages of ACO are positive feedback for recovery of good solutions, distributed computation, which avoids premature convergence. It has been mainly used in finding the shortest route in transmission network, short term generation scheduling and optimal unit commitment. PSO can be used to solve complex optimization problems, which are non-linear, no differentiable and multi-model. The main merits of PSO are its fast convergence speed and it can be realized simply for less parameters need adjusting. PSO has been mainly used to solve Bi-objective generation scheduling, optimal reactive power dispatch and to minimize total cost of power generation.

4 Particle swarm optimization

It is based on the ideas of social behavior of organisms such as animal flocking and fish schooling. H. Yoshida proposed a Particle Swarm Optimization (PSO) for reactive power and Voltage/VAR Control (VVC) considering voltage security assessment. It determines an online VVC strategy with continuous and discrete control variables such as AVR operating values of generators, tap positions of OLTC of transformers and the number of reactive power compensation equipment[6].

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) [7] This value is called pbest. Another “best” value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called lbest. When a particle takes all the population as its
topological neighbors, the best value is a global best and is called gbest. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its pbest and lbest locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and lbest locations.

4.1 Implementation of PSO for OPF problem
A swarm consists of a set of particles moving within the search space, each representing a potential solution (fitness). In a physical n-dimensional search space, the position and velocity of each particle are represented as the vectors \( X_i = (x_1, \ldots, x_n) \) and \( V_i = (v_1, \ldots, v_n) \), respectively.

Let \( P_{\text{best}_i} = (b_1, \ldots, b_n) \) and \( G_{\text{best}_i} = (g_1, \ldots, g_n) \) be the position of the individual \( i \) and its neighbor’s best position so far. Using this information, the modified velocity of each individual can be calculated using the following equation and the distance from \( P_{\text{best}} \) and \( G_{\text{best}} \) as shown in

\[
V_i^{k+1} = \omega V_i^k + c_1 r_1 (P_{\text{best}_i} - X_i^k) + c_2 r_2 (G_{\text{best}_i} - X_i^k)
\]

where \( V_i^k \) is current velocity of individual \( i \) at iteration \( k \), \( k+1 \) modified velocity of individual \( i \) at iteration \( k+1 \), \( X_i^k \) is the current position of individual \( i \) at iteration \( k \), \( X_i^k \) is the current position of individual \( i \) at iteration \( k \), \( \omega \) is the inertia weight parameter, \( c_1 \) and \( c_2 \) are the acceleration factors, \( r_1 \) and \( r_2 \) are random numbers between 0 and 1, \( P_{\text{best}_i} \) is the personal best position of individual \( i \) until iteration \( k \), \( G_{\text{best}_i} \) is the global best position of the group until iteration \( k \).

Each individual moves from the current position to the next one by the modified velocity using the following equation:

\[
X_i^{k+1} = X_i^k + V_i^{k+1}.
\]

The parameters \( c_1 \) and \( c_2 \) are set to constant values. Low values allow the individual to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement towards target regions. Hence the acceleration constants \( c_1 \) and \( c_2 \) are normally set as 2.0 whereas \( r_1 \) and \( r_2 \) are random values, and they are uniformly distributed between zero and one. These values are not the same for each iteration because they are generated randomly every time.

4.1.1 PSO algorithm
The technique is initialized with a population of random solutions or particles and then searches the optima by updating generations. Each individual particle \( i \) has the following three properties: a current position in search space \( x_i \), a current velocity \( v_i \), and a personal best position in search space \( y_i \).

In every iteration, each particle is updated by the following two best values. The first one is the personal best position \( y_i \) which is the position of the particle \( i \) in the search space, where it has reached the best solution so far. The second one is the global best solution \( y* \) which is the position yielding the best solution among all the \( y_i \)'s. The pbest and gbest values are updated at time \( t \) using the equations 1 and 2.

Here it is assumed that the swarm has \( s \) particles. Therefore, \( i \in 1, \ldots, s \) and assuming the minimization of the objective function \( F_i \),

\[
Y(t+1) = \begin{cases} 
  y_i(t+1), & \text{if } f(y_i(t+1)) \leq f(x_i(t+1)) \\
  y_i(t), & \text{otherwise}
\end{cases}
\]

(6)

After finding the two best values, each particle updates its velocity and current position. The velocity of the particle is updated according to its own previous best position and the previous best position of its companions which is given .This new velocity is added to the current position of the particle to obtain its next position.
The acceleration coefficients control the distance moved by a particle in the iteration. The inertia weight controls the convergence behavior of PSO. Initially the inertia weight was considered as a constant value.

However, experimental results indicated that it is better to initially set the inertia weight to larger value and gradually reduce it to get refined solutions. A new inertia weight which is neither set to a constant value nor set as a linearly decreasing time-varying function is used in this paper.

5 Seeker Optimization Algorithm

The SOA is based on the concept of simulating the act of human searching, where the search direction is based on the empirical gradient by evaluating the response to the position changes and the step length is based on uncertainty reasoning by using a simple Fuzzy rule[2].

SOA operates on a set of solutions called search population. The individual of this population is called seeker[2]. In order to add a social component for social sharing of information, a neighborhood is defined for each seeker. In the present simulations, the population is randomly categorized into K=3 subpopulations in order to search over several different domains of the search space and all seekers in the same subpopulation constitute a neighborhood. Assume that the optimization problems to be solved are minimization problems.

The main characteristics features of this algorithm are the following:

- The algorithm uses search direction and step length to update the positions of seekers.
- The calculations of search direction are based on a compromise among egotistic behavior, altruistic behavior and pro-activeness behavior.
- Fuzzy reasoning is used to generate the step length because the uncertain reasoning of human searching could be the best described by natural linguistic variables and a simple if else control rule.

The search direction $d_i(t)$ and a step length $\alpha_i(t)$ are computed separately for each seeker $i$ on each dimension $j$ for each time step $t$ where $\alpha_i(t)\geq 0$ and $d_i(t)\in\{-1,0,1\}$.

$d_i(t)=1$ means the $i$-th seeker goes towards the positive direction of the coordinate axis on the dimension $j$.

$d_i(t)=-1$ means the seeker goes towards the negative direction and

$d_i(t)=0$ means the seeker stays at the current position.

For each seeker $i$ ($1 \leq i \leq s$, $s$ is the population size), position update on each dimension $j$ ($1 \leq j \leq D$) is given by the following:

$$X_{ij}(t+1) = X_{ij}(t) + \alpha_i(t) \cdot d_i(t)$$ (8)

Since the subpopulations are searching using their own informations, they are easy to converge to a local optimum[2].

In order to avoid this situation the positions of the worst $k=1$ seekers of each subpopulation are combined with the best one in each of the other $k-1$ subpopulations using the following binomial crossover operator:

$$x_{knj,\text{worst}} = \begin{cases} x_{lj,\text{best}} & \text{if } R \leq 0.5 \\ x_{knj,\text{worst}} & \text{else} \end{cases}$$ (9)

where $R_1$ is a uniformly random real number within $[0, 1]$. $x_{knj,\text{worst}}$ is denoted as the $j$-th dimension of the $n$-th worst position in the $K$-th subpopulation, $X_{lj,\text{best}}$ is the $j$-th dimension of the best position in the $l$-th subpopulation with $n,k,l=1,2,\ldots,k-1$ and $k\neq l$.

5.2 SOA Flow chart and Algorithm

The basic form of the proposed SOA algorithm can only handle continuous variables. However, both tap position of transformations and reactive power source installations are discrete or integer variables in optimal reactive power dispatch problem. To handle integer variables without any effect on the implementation of SOA, the seekers will still search in a continuous space regardless of the variable type, and then truncating the corresponding dimensions of the seekers’ real-value positions into the integers is only performed in evaluating the objective function. The reactive power optimization based on SOA can be described as follows.

Step 1) Read the parameters of power system and the proposed algorithm, and specify the lower and upper limits of each variable.

Step 2) Initialize the positions of the seekers in the search space Randomly and uniformly. Set the time step $t=0$

Step 3) Calculate the fitness values of the initial positions using the objective function based on
Step 4) Let $t = t + 1$

Step 5) Select the neighbors of each seeker.

Step 6) Determine the search direction and step length for each seeker, and update his position.

Step 7) Calculate the fitness value of the new positions using the objective function based on the Newton–Raphson power flow analysis results. Update the historical best position among the population and the historical best position of each seeker.

Step 8) Go to Step 4 until a stopping criterion is satisfied.

6 Simulation techniques and results

In particle swarm optimization Default population size: 24
Number of particles used: 20
Number of iterations set: 30
Initial inertia weight: 0.9 and final inertia weight: 0.4
Pso seed, default=0
  = 0 for initial positions all random
  = 1 for initial particles as user input

The table shows, variation of global best value of the generation for the number of iterations. In the end of all iterations the generation of all plants is maximum with reduced losses. And the bus voltages in each bus in the end of iteration are given. This graph depicts the variation of voltage in each bus. The bus voltage drops due to the losses in the bus. This problem derived for without considering the constrains and simulation results are produced in 10 seconds.
Table 1: PSO output

<table>
<thead>
<tr>
<th>Bus no</th>
<th>Voltage</th>
<th>Bus no</th>
<th>Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0600</td>
<td>16</td>
<td>1.0471</td>
</tr>
<tr>
<td>2</td>
<td>1.0430</td>
<td>17</td>
<td>1.0415</td>
</tr>
<tr>
<td>3</td>
<td>1.0254</td>
<td>18</td>
<td>1.0304</td>
</tr>
<tr>
<td>4</td>
<td>1.0171</td>
<td>19</td>
<td>1.0277</td>
</tr>
<tr>
<td>5</td>
<td>1.0100</td>
<td>20</td>
<td>1.0317</td>
</tr>
<tr>
<td>6</td>
<td>1.0148</td>
<td>21</td>
<td>1.0345</td>
</tr>
<tr>
<td>7</td>
<td>1.0050</td>
<td>22</td>
<td>1.0350</td>
</tr>
<tr>
<td>8</td>
<td>1.0100</td>
<td>23</td>
<td>1.0296</td>
</tr>
<tr>
<td>9</td>
<td>1.0530</td>
<td>24</td>
<td>1.0237</td>
</tr>
<tr>
<td>10</td>
<td>1.0467</td>
<td>25</td>
<td>1.0204</td>
</tr>
<tr>
<td>11</td>
<td>1.0820</td>
<td>26</td>
<td>1.0027</td>
</tr>
<tr>
<td>12</td>
<td>1.0599</td>
<td>27</td>
<td>1.0269</td>
</tr>
<tr>
<td>13</td>
<td>1.0710</td>
<td>28</td>
<td>1.0128</td>
</tr>
<tr>
<td>14</td>
<td>1.0450</td>
<td>29</td>
<td>1.0071</td>
</tr>
<tr>
<td>15</td>
<td>1.0402</td>
<td>30</td>
<td>0.9957</td>
</tr>
</tbody>
</table>

Table 1: PSO output

6.1 Simulation Output of SOA

Seeker optimization algorithm for optimal power flow is similar to the Particle swarm optimization where as it is based on behavior of human searching. Although PSO is easy to implement, the performance of PSO also depends on its parameters and may be influenced by premature convergence and stagnation problem. So the disadvantage of PSO algorithm is overcome and best result for optimal power flow problem is obtained by seeker optimization algorithm.

Table 2: SOA without capacitor

<table>
<thead>
<tr>
<th>Bus no</th>
<th>V pu</th>
<th>Angle degree</th>
<th>Bus no</th>
<th>V pu</th>
<th>Angle degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0600</td>
<td>0.0000</td>
<td>16</td>
<td>0.9810</td>
<td>-15.9998</td>
</tr>
<tr>
<td>2</td>
<td>1.0130</td>
<td>-4.9686</td>
<td>17</td>
<td>0.9704</td>
<td>-16.2878</td>
</tr>
<tr>
<td>3</td>
<td>0.9947</td>
<td>-7.4130</td>
<td>18</td>
<td>0.9618</td>
<td>-17.0910</td>
</tr>
<tr>
<td>4</td>
<td>0.9797</td>
<td>-9.1675</td>
<td>19</td>
<td>0.9572</td>
<td>-17.2733</td>
</tr>
<tr>
<td>5</td>
<td>0.9600</td>
<td>-14.1797</td>
<td>20</td>
<td>0.9605</td>
<td>-17.0383</td>
</tr>
<tr>
<td>6</td>
<td>0.9757</td>
<td>-11.0397</td>
<td>21</td>
<td>0.9597</td>
<td>-16.6744</td>
</tr>
<tr>
<td>7</td>
<td>0.9611</td>
<td>-12.9026</td>
<td>22</td>
<td>0.9646</td>
<td>-16.4171</td>
</tr>
<tr>
<td>8</td>
<td>0.9800</td>
<td>-11.9201</td>
<td>23</td>
<td>0.9600</td>
<td>-16.6855</td>
</tr>
<tr>
<td>9</td>
<td>0.9918</td>
<td>-14.3072</td>
<td>24</td>
<td>0.9519</td>
<td>-16.7732</td>
</tr>
<tr>
<td>10</td>
<td>0.9740</td>
<td>-16.0766</td>
<td>25</td>
<td>0.9599</td>
<td>-16.5460</td>
</tr>
<tr>
<td>11</td>
<td>1.0220</td>
<td>-14.3072</td>
<td>26</td>
<td>0.9411</td>
<td>-17.0184</td>
</tr>
<tr>
<td>12</td>
<td>0.9995</td>
<td>-15.4241</td>
<td>27</td>
<td>0.9741</td>
<td>-16.1023</td>
</tr>
<tr>
<td>13</td>
<td>1.0110</td>
<td>-15.4241</td>
<td>28</td>
<td>0.9738</td>
<td>-11.7559</td>
</tr>
<tr>
<td>14</td>
<td>0.9818</td>
<td>-16.3934</td>
<td>29</td>
<td>0.9531</td>
<td>-17.4624</td>
</tr>
<tr>
<td>15</td>
<td>0.9752</td>
<td>-16.4195</td>
<td>30</td>
<td>0.9410</td>
<td>-18.4421</td>
</tr>
</tbody>
</table>

Fig 4: SOA voltage variations

Fig 5: SOA voltage variations
Angle algorithm has it and particle ion M. F. AlHajri, using it n act Thst proposed W react in Z. Ren, in simple to system dispatch problem are particle intelligent better v in intelig Syst. 1 human additional Y Zhu, and Xu used quickly and to this modified genetic stochastic the to dispatch o W chaotic s is of that W S Krost, optimization s to v Intelli R. optimization 42nd v application T. approach techniques pp. applied and and benefits of S Meeting J heuristic on proposed, a and identification simulation results problems. Herrera, be, easy balancing global Eng. ‘A Applications di based ability react P optimization Planning and Y. Chaohua to GA based reactive power dispatch stabilit a a the a a the no. perfor in and and based and capacitor S Santos K. Bakare, simulating 2007, 54, t trends pp.3234 considering Y. Santos of Multi-type FACTS Devices in a Power System by Means of Genetic Algorithm’, IEEE Trans on Power System, Vol.16, No.3, pp.658-667.


7 Conclusion
SOA is a novel heuristic stochastic optimization algorithm based on simulating the act of human searching. The algorithm has the additional advantage of being easy to understand, simple to implement so that it can be used for a wide variety of design and optimization tasks. In this paper, a SOA-based reactive power optimization approach is proposed, and the benefits of SOA for optimal reactive power dispatch problem are studied. The simulation results show that SOA has better performance in balancing global search ability and convergence speed than other algorithms. So, it is believed that the proposed SOA approach is capable of quickly and effectively solving reactive power dispatch problem and will become a promising candidate for the OPF problems.

GA is used for optimal placement of FACTS devices that GA based reactive power dispatch algorithm is able to minimize the power loss in the system. Overall performance of the SOA-GA algorithm and its advantages were discussed and the algorithm shows that it is easy to implement and also produces good results for optimal power flow problem than other algorithms.

References:

<table>
<thead>
<tr>
<th>Bus no</th>
<th>V pu</th>
<th>Angle degree</th>
<th>Bus no</th>
<th>V pu</th>
<th>Angle degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0600</td>
<td>0.0000</td>
<td>16</td>
<td>1.0315</td>
<td>-15.5967</td>
</tr>
<tr>
<td>2</td>
<td>1.0430</td>
<td>-5.3525</td>
<td>17</td>
<td>1.0202</td>
<td>-15.8422</td>
</tr>
<tr>
<td>3</td>
<td>1.0200</td>
<td>-7.5318</td>
<td>18</td>
<td>1.0128</td>
<td>-16.5781</td>
</tr>
<tr>
<td>4</td>
<td>1.0108</td>
<td>-9.2848</td>
<td>19</td>
<td>1.0080</td>
<td>-16.7371</td>
</tr>
<tr>
<td>5</td>
<td>1.0100</td>
<td>-14.1692</td>
<td>20</td>
<td>1.0110</td>
<td>-16.5223</td>
</tr>
<tr>
<td>6</td>
<td>1.0101</td>
<td>-11.0627</td>
<td>21</td>
<td>1.0102</td>
<td>-16.1952</td>
</tr>
<tr>
<td>7</td>
<td>1.0023</td>
<td>-12.8655</td>
<td>22</td>
<td>1.0147</td>
<td>-15.9655</td>
</tr>
<tr>
<td>8</td>
<td>1.0100</td>
<td>-11.8168</td>
<td>23</td>
<td>1.0106</td>
<td>-16.2084</td>
</tr>
<tr>
<td>9</td>
<td>1.0402</td>
<td>-14.0508</td>
<td>24</td>
<td>1.0033</td>
<td>-16.3066</td>
</tr>
<tr>
<td>10</td>
<td>1.0231</td>
<td>-15.6455</td>
<td>25</td>
<td>1.0125</td>
<td>-16.1605</td>
</tr>
<tr>
<td>11</td>
<td>1.0820</td>
<td>-14.0508</td>
<td>26</td>
<td>1.0000</td>
<td>-16.7794</td>
</tr>
<tr>
<td>12</td>
<td>1.0503</td>
<td>-15.0935</td>
<td>27</td>
<td>1.0240</td>
<td>-15.7112</td>
</tr>
<tr>
<td>13</td>
<td>1.0710</td>
<td>-15.0935</td>
<td>28</td>
<td>1.0091</td>
<td>-11.7434</td>
</tr>
<tr>
<td>14</td>
<td>1.0330</td>
<td>-15.9691</td>
<td>29</td>
<td>1.0076</td>
<td>-17.0359</td>
</tr>
<tr>
<td>15</td>
<td>1.0263</td>
<td>-15.9809</td>
<td>30</td>
<td>1.0000</td>
<td>-18.0205</td>
</tr>
</tbody>
</table>

Table 3: SOA with capacitor


