PARALLEL CONTINGENCY ANALYSIS USING DIFFERENTIAL EVOLUTION BASED SOLUTION FOR BRANCH OUTAGE PROBLEM

Oğuzhan Ceylan
Informatics Institute,
Istanbul Technical University,
Istanbul, Turkey
Email: oguzhan@be.itu.edu.tr

Hasan Dağ
Information Technologies Department,
Kadir Has University,
Istanbul, Turkey
Email: hasan.dag@khas.edu.tr

Aydoğan Özdemir
Electrical-Electronics Faculty,
Istanbul Technical University,
Istanbul, Turkey
Email: ozdemir@elk.itu.edu.tr

ABSTRACT

Contingency analysis is one of the most fundamental work an electricity management center operator has to perform regularly. If both bus voltage magnitudes and reactive power flowing on the branches during any type of outages are within the acceptable limits, the system is called secure. In this paper we solve the contingency problem using a recently developed local constrained optimization based branch outage problem. The optimization problem resulted from the formulation of branch outage is solved by differential evolution method. Using Matlab's parallel computing toolbox, contingency analysis for IEEE 300 test system is performed and the results are presented. The study shows that it is straight forward to implement contingency analysis on the Matlab's parallel environment and obtain near linear speedups.

1. INTRODUCTION

Electrical power system is one of the most complex nonlinear system created by human being. Operation of it in a secure way require regular testing for unexpected events, such as contingency analysis for line outages to assess the system's security. In order to decide whether a system is secure or not, all of the branch outages in an electrical power system are simulated. This process is a time consuming one especially in the large power systems and it should be repeated as frequently as possible. Hence, parallel simulation of branch outages will enable operator to asses system security w.r.t line outages faster and as frequent as he/she wishes.

One of the recent study [1] formulates branch outage problem as a local constrained optimization problem, where the outaged branch is replaced by fictitious sources. This method has advantages compared to full AC load flow based methods both in accuracy and speed. There are also DC power flow based methods [2] but they suffer especially from voltage magnitudes and reactive power flows.

Electrical power system application of intelligent methods such as genetic algorithms, simulated annealing method, particle swarm optimization, and differential evolution method have gained popularity in recent years. These methods initially form a group of random solutions in some specified variable ranges, then by some operations, such as crossover and mutation, new solution candidates are formed and compared to the old ones in terms objective function fitness. If the newly computed fitness is better than the old one the new candidate replaces the old one.

Differential evolution method seems to be one of the fastest intelligent methods for optimization problems. It has been applied to several power system problems, such as, power system planning [3], transient stability constrained optimal power flow [4], unit commitment [5], reactive power optimization [6], and economic dispatch [7]. In this paper we present the results of application of differential evolution method to branch outage problem within the contingency analysis study in a parallel environment.

The rest of the paper is organized as follows. In the second part branch outage model used in this paper is presented. In the third part the details of differential evolution method are given. Next, the algorithm for both serial and parallel contingency analysis is given. In the tests section speedup and cpu time graphics for IEEE-300 bus test case are given. We finally present some conclusions.

2. BRANCH OUTAGE PROBLEM

In branch outage problem the main attention is given to the outaged branch. Reactive power flowing through, the transferred reactive power, and reactive power loss on the outaged branch can be represented as Q_{ij}, Q_{ij}^T, and Q_{Li} respectively. These reactive powers can be expressed in terms of system variables as follows.

$$\begin{aligned} Q_{ij} = &-[V_i^2 - V_iV_jcos\delta_{ji}]b_{ij} + V_iV_jg_{ij}\sin\delta_{ji} \\ &- V_i^2\frac{b_{i0}}{2} \end{aligned} \tag{1}$$

$$Q_{ij}^T = -[V_i^2 - V_j^2]\frac{b_{ij}}{2} + V_iV_jg_{ij}sin\delta ji \tag{2}$$

9781-4244-3428-2/09/$25.00 ©2009 IEEE

$$Q_{Li} = -[V_i^2 + V_j^2 - 2V_iV_j \cos \delta ji]\frac{b_{ij}}{2} - (V_i^2 + V_j^2)\frac{b_{i0}}{4} \tag{3}$$

In the model [1], the case of an outage is represented by two fictitious sources: one in the sending end and one in the receiving end of the outaged line.

The procedure for the existing method can be given as follows.

1. Select an outage of a branch connected between busses i and j, and number it as k,

2. Calculate bus voltage phase angles using linearized MW flows (see [2] for details),

$$\delta_l = \delta_l - (X_{li} - X_{lj}) \triangle P_k, \quad l = 2, 3, \cdots, \text{NB} \tag{4}$$

$$\triangle P_k = \frac{P_{ij}}{\frac{1-(X_{ii}+X_{jj}-2X_{ij})}{x_k}} \tag{5}$$

   where, X represents the inverse of the bus susceptance matrix, P_{ij} is the pre-outage active power flow through the outaged line, and x_k represents the reactance of the line at hand. If the voltage magnitudes are calculated, then the calculation of the busses included in the bounded network would suffice.

3. Calculate the reactive power transfer $\overline{Q}_{ij}^T$between the busses. This power includes the increment due to the change in bus voltage phase angles.

4. Minimize the reactive power mismatches of busses i and j. This process is mathematically equivalent to the following constrained optimization problem.

$$\begin{aligned} \min_{wrt\ Q_{si},Q_{sj}} \quad & \| \, Q_i - (\overline{Q}_{ij} + \overline{Q}_{Li}) + Q_{Di} \\ & Q_j - (-\overline{Q}_{ij} + \overline{Q}_{Li}) + Q_{Dj} \, \| \\ \text{subject to} \quad & g_q(V_b) = \triangle Q_b - B_b \triangle V_b = 0 \end{aligned} \tag{6}$$

   where, $\| \cdot \|$ is the Euclidean norm of a vector. Equation (6) is linear reactive power equation for load busses, $\triangle Q$ is reactive power mismatch vector, V is bus voltage magnitude vector and B is bus susceptance matrix. It should be stated that only two elements of $\triangle Q$ vector are nonzero, and they are represented as below.

$$[\triangle Q] : [\triangle Q]_i = -[\triangle Q]_j = Q_{si} - Q_{ij}. \tag{7}$$

   On the other hand, subscript b is used to denote the bounded region where the optimization process is done.

3. DIFFERENTIAL EVOLUTION

Differential evolution is a stochastic method, which is derived from genetic algorithms by Storn and Price [8]. It has similar operators as those of the other evolutionary algorithms, such as crossover, mutation, and selection.

The steps of a general differential evolution algorithm can be summarized as below:

1. Initialize population: A population consisting of Np randomly created vectors is generated. Np is a user specified value and is generally selected as 10-15 times of the number of unknowns. Size of each vector in a population is equal to the number of unknowns. Representation of a population is given below.

$$P^{(G)} = [x_1^{(G)}, \cdots, x_{Np}^{(G)}] \tag{8}$$

   Random initialization of a vector in a population can be performed as,

$$x_i^{(G)} = x_{i(L)} + rand_i[0,1](x_{i(H)} - x_{i(L)}) \tag{9}$$

   where, $x_{i(L)}$ and $x_{i(H)}$ are the lower and upper bounds for the initial vectors to be constructed in the first population.

2. Add a weighted difference vector between the two individual vectors to the third one and create Np new mutant vectors. This process is performed for all vectors in a population and can be expressed as,

$$x_i^{'(G)} = x_{r3}^{(G)} + F(x_{r1}^{(G)} - x_{r2}^{G}) \tag{10}$$

   where, $i \neq r_1 \neq r_2 \neq r_3$ and r_1, r_2 and r_3 are randomly selected numbers from 1 to Np, F is a real constant positive scaling factor chosen within the range (0,2] and $x_i^{'(G)}$ represents the created mutant vector [9].

3. Some of the newly generated mutant vectors in the second step are used to generate trial vectors and some of them will remain unchanged in trial vectors, regarding to a constant number called crossover constant or a random parameter q chosen once for each i. Crossover constant (CR) is chosen within the range [0,1], and for all i parameters it is compared with a randomly generated number between 0 to 1 inclusive. The following expression defines the crossover process.

$$x_i^{\text{trial}(G)} = \begin{cases} x_{ji}^{'(G)} & \text{if rand}(0,1) \leq (CR) \text{ or } j = q, \\ x_{ji}^{(G)} & \text{otherwise.} \end{cases} \tag{11}$$

4. In selection step, the algorithm decides whether or not the trial vector will be part of the next generation. This decision is done by comparing the fitness values of the

trial vectors with the associated target vectors. This process can be represented as below.

$$x_i^{(G+1)} = \begin{cases} x_i^{\text{trial}(G)} & \text{if} f(x_i^{\text{trial}(G)}) \leq f(x_i^{(G)}), \\ x_i^{(G)} & \text{otherwise.} \end{cases} \quad (12)$$

5. If a predetermined stopping criterion is met, stop, otherwise go to step 2.

4. APPLICATION OF DIFFERENTIAL EVOLUTION ALGORITHM TO BRANCH OUTAGE PROBLEM

Differential evolution algorithm for solving the local constrained optimization problem encountered in branch outage problem can be outlined as follows [10]:

1. Run a power flow for pre-outage state and obtain initial load bus voltage magnitudes in the bounded region.

2. By choosing upper bound and lower bound for Q_{si}, initial $\overline{Q}_{ij}^{T} + 1$ $\overline{Q}_{ij}^{T} - 1$ respectively, choose Np different Q_{si} values. Using (7), form $\triangle Q$ vectors and after solving the equation below, update the voltage magnitudes.

$$(B_b)^{-1} \triangle Q_b = \triangle V_b \quad (13)$$

3. For all vectors in the population create Np new mutant Q_{si} vectors using (10).

4. Generate Q_{si} trial vectors using (11).

5. In order to determine the existence of the trial vector in the next generation, evaluate the fitness function using (6), and form the new generation.

6. If Euclidean norm of the objective function to be minimized is smaller than a predetermined value for the best element in the generation, or the maximum number of iterations is reached to the limit stop, otherwise go to step 2.

5. CONTINGENCY ANALYSIS

All branches, but the ones connected to a generator in both ends, are considered in contingency analysis. The algorithm starts with running a base case load flow. After having obtained initial voltage magnitudes and angles once for all outage scenarios, a vector BR, that has indices of branches that will be outaged is formed. Assuming that BR has N elements and the contingency analysis will be performed on P processors, following arithmetical operation is performed;

$$\text{Number of elements per processor} = N/P \quad (14)$$

Then a new matrix $PROCS$ with N/P elements in each column and P rows is formed. Each N/P of N elements of BR vector is inserted in each row of $PROCS$ matrix consecutively. If (14) has a remainder Rem, Rem number of elements of BR vector are distributed to each row of $PROCS$ matrix one by one in turn in order to achieve a good load balance. It should be stated that if there is a remainder of (14) then, the first Rem rows of the matrix $PROCS$ will have one more element than the remaining $P - Rem$ rows.

Using spmd (single program multiple data) property of Matlab Parallel Computing Toolbox, each row of the matrix $PROCS$ is given to differential evolution program as input. In other words, each processor runs differential evolution based branch outage subprogram, N/P times or $N/P + 1$ times regarding to the corresponding row of $PROCS$ matrix.

Parallel contingency algorithm can be summarized as follows.

1. Run a base case load flow in order to obtain initial voltage magnitudes and angles.

2. By excluding the branches connected to generators at both ends, form a vector BR that has indicies of branches that will be outaged.

3. By dividing the number of elements of BR to P, obtain outages per processor as given in (14). If the remainder of this operation is not zero, than distribute remaining outages one by one to each processor in turn as described above.

4. For each processor

   - Run a differential evolution based branch outage solution algorithm.

6. TEST RESULTS

IEEE-300 Bus test system is used for contingency analysis. By excluding the branches connected to generators in both ends, 403 cases have been simulated. Matlab based open-source software Matpower [11] and the parellel computing toolbox of Matlab [12] are used as tools. All programs are written in Matlab. Programs are run on a distributed cluster, with 37 nodes, each consisting of 2 CPU's having 3.40 GHz CPU, and 2 GB memory.

Differential evolution parameters are chosen as follows: $Np = 30$, $CR = 0.9$, $F = 0.8$. If two consecutive fitness functions have difference less than 0.01 then the algorithm stops.

Figure (1) shows relative speed up versus number of processors. At most 40 processors are used in simulations, since it is observed that increasing number of processors does not provide additional speedup On the other hand (2) shows cpu time versus number of processors. It could easily be seen from the figures that by using 40 processors, up to 25 times faster results can be obtained.

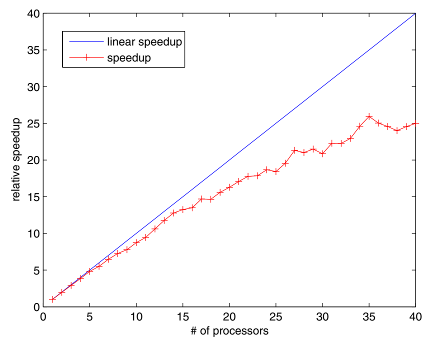

Fig. 1: Relative speedup versus number of processors.

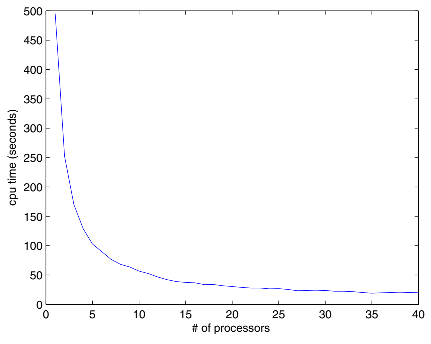

Fig. 2: Cpu time versus number of processors.

7. CONCLUSION

In this study, contingency analysis using the constrained optimization problem representing the line outage phenomena in an electrical power system is performed in parallel using parallel computing toolbox of Matlab. Constrained optimization problem encountered in branch outage problem is solved by differential evolution method.

Simulation results have shown that the problem at hand can be easily parallelized and reasonable speed up values can be obtained. For IEEE-300 bus test system for 40 processors relative speedup achieved is 25. Parallelization has been performed in Matlab, which could be a good alternative especially if multi-core personal computers are used in terms of cost and speed.

8. REFERENCES

[1] A. Ozdemir, Y. J. Lim, and C. Singh, "Branch outage simulation for MWAR flows: bounded network solution", *IEEE Trans. Power Syst.*, vol. 18, pp. 1523–1528, Nov. 2003.

[2] A. J. Wood and B. F. Wollenberg, *Power Generation Operation and Control*, 2nd ed., New York, USA: Wiley, 1998.

[3] Y. G. Yang, Z. Y. Dong, and K. P. Wong, "A modified differential evolution algorithm with fitness sharing for power system planning", *IEEE Trans. Power Syst.*, vol. 23, pp. 514–522, May 2008.

[4] H. R. Cai, Y. C. Chung, and K. P. Wong, "Application of differential evolution algorithm for transient stability constrained optimal power flow", *IEEE Trans. Power Syst.*, vol. 23, pp. 719–728, May 2008.

[5] S. Patra, S. K. Gowami, and B. Goswami, "A binary differential evolution algorithm for transmission and voltage constrained unit commitment", *Joint International Conference on Power System Technology and IEEE Power India Conference POWERCON 2008*, pp. 1–8, Oct. 2008.

[6] X. Zhang, W. Chen, C. Dai, and A. Guo, "Self adaptive differential evolution algorithm for reactive power optimization", *Fourth International Conference on Natural Computation 2008. ICNC '08*, vol.6, pp. 560–564, Oct. 2008.

[7] R. E. Perez-Guerrero, and J. R. Cedeno-Maldonado, "Economic power dispatch with non-smooth functions using differential evolution", *Proceedings of the 37th Annual North American Power Symposium 2005*, pp. 183–190, Oct. 2005.

[8] R. Storn, and K. Price, "Differential Evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, ICSI, http://http.icsi.berkeley.edu/ storn/litera.html.

[9] K. V. Price, "Differential evolution: a fast and simple numerical optimizer", *Biennial Conference of the North American Fuzzy Information Processing Society*, pp. 524–527, June 1996.

[10] O. Ceylan, A. Ozdemir, and H. Dag, "Application of differential evolution method to branch outage problem", The 2009 International Conference on Genetic and Evolutionary Methods (GEM 2009), Las Vegas, USA, July 2009.

[11] R. Zimmermann, and D. Gan, "Matpower Manual", *USA: PSERC*, Cornell Univ. 1997.

[12] "The mathworks distributed computing toolbox and matlab distributed computing engine" 2.0.1 http:www.mathworks.com/products/distribtb