Power Systems Dynamic Security Enhancement by the use of Efficient Heuristics

Carlos A. S. Neto[1] **Marcus Th. Schilling**[1] **Júlio C. S. de Souza**[1] **Luiz Satoru Ochi**[2]
cneto@ic.uff.br schilling@ic.uff.br julio@ic.uff.br satoru@ic.uff.br

1 Electrical Engineering Department, Fluminense Federal University, Niterói, RJ, BR
2 Computer Science Department, Fluminense Federal University, Niterói, RJ, BR

Keywords: Dynamic Security Assessment, Evolutionary Algorithms, GRASP.

Abstract

Dynamic security corresponds to the capacity of the electrical system to keep synchronism among all generators, in the event of a contingency. Usually, the optimal dynamic security problem is solved by specialists, who investigate the worst scenarios and contingencies using conventional simulation tools. This work shows the main features of the implementation of a new approach, based on the combination of a complete time domain simulation and metaheuristics. The fitness function corresponds to the stability margin, which is computed for each machine and for each contingency. Due to its innovativeness, two different implementations will be tested. One based on an evolutive algorithm and the other based on a GRASP algorithm. Tests were done with the Anderson & Fouad test case [Fouad 77] which and the IEEE 14 buses test case.

1 Introduction

The power systems dynamic security optimization problem is associated with the safety measures needed to prevent any equipment stress in the operation of an electrical network. Stressed equipments may be switched off by their protective relays and it may result in a cascade of events causing a partial or a major blackout. Generation active power dispatch and network voltage profile are the main control variables that can be adjusted to guarantee a safe steady state operation, under normal condition, or in the event of a set of credible contingencies. More recently a new concern has arisen, regarding the dynamic response of the system, once the post-contingency steady state may not show the poor damped oscillations that may cause a system collapse. They result from the electromechanical interaction of the massive rotating generators. Therefore, power systems dynamic security seeks for the safe operation of the electrical network, without any equipment stress and with the capacity of keeping synchronism among all generators, under normal and faulted conditions.

The dynamic security problem is usually solved by specialists, who investigate the worst scenarios using conventional simulation tools. It is quite likely that the solution provided by such conservative procedure is not the best solution. This work proposes an approach considering more realistic premises and based on intelligent optimization techniques. There are already some initiatives in this area that can be classified basically in a twofold family of algorithms. The first one corresponds to the use of optimal power flow [Kuo 95], [Sun 04] and [Yuan 03], while the second one is based on the use of artificial intelligence or fuzzy logic [Edwards 96], [Groom 96], [Kamwa 01], [Pao 92], [Sobajic 89] and [Tso 95]. Dynamic modeling is the major drawback in the first approach, while the proper definition of the training set and input variables is the biggest challenge in the second. So far, there is no implementation that makes use of a full dynamic model considering all variables present in the power systems (generation dispatch, voltage profile, network topology, etc…).

The motivation of this work is to explore the generalization capacity, provided by metaheuristics, to allow for the use of complete time domain simulations in the fitness function evaluation. This means a more reliable approach, once there is no simplification, i.e. all system dynamics are considered. The fitness function output corresponds to the stability margin, which is computed for each generator and for each contingency. This margin indicates how far from loosing synchronism a generator is. The fitness function corresponds to the smallest margin considering all generators and all contingencies. The whole system is unstable if, at least, one generator looses synchronism. Thus, a bigger margin indicates a safer system and a negative margin indicates that the system is unstable.

Two different metaheuristics are used to check the feasibility of this innovative implementation. The first metaheuristic is based on an arithmetic evolutionary algorithm called LUDE (Line Up Differential Evolution), where all solutions of each generation are lined up, according to the fitness function value. The position in the list defines the likelihood of changes due to the crossover and mutation operators. Another important aspect of this algorithm is that it has only two parameters to be defined: the population size and the number of iterations.

The GRASP algorithm was implemented under the premises that systems with bigger total inertia tend to be more stable. Thus, the algorithm prioritizes the dispatch of generators with bigger inertia.

Next section contains the formal description of power system dynamic security optimization, along with the computation of stability margin. Sections 3 and 4 contain the descriptions of the evolutive algorithm called LUDE and the GRASP algorithm respectively. Results are presented and discussed in section 5 and conclusions are drawn in section 6.

2 Power System Dynamic Security Optimization and Stability Margin

Power system dynamic security optimization problems can be used with different purposes like project of control devices or definition of generation dispatch, for example. The difference resides on the set of variables to be controlled. On both approaches the objective function is to find the safest solution, such that the system survives the set of credible contingencies, with no violation and without loosing synchronism.

Despite generation dispatch being a continuous variable, this optimization problem has a combinatorial nature due to the decision of which generators to dispatch and the value of each generator dispatch. The formulation of this problem is not trivial, but could be defined as the maximization of the stability index, considering physical and operative restrictions like equipments capabilities and voltage ranges. A formal description could be as follows:

$$\begin{array}{llll}
\text{Maximize} & \boldsymbol{Min(Margin_i^j)} & \boldsymbol{\forall i=1, NGen,\ \forall j=1, NCtg} & (1) \\
\text{Subject to:} & \boldsymbol{Margin_i^j > 0} & \boldsymbol{\forall i=1, NGen,\ \forall j=1, NCtg} & (2) \\
 & \boldsymbol{Pg_{Min\,i} \le Pg_i^j \le Pg_{Maxi}} & \boldsymbol{\forall i=1, NGen,\ \forall j=1, NCtg} & (3) \\
 & \boldsymbol{Qg_{Min\,i} \le Qg_i^j \le Qg_{Maxi}} & \boldsymbol{\forall i=1, NGen,\ \forall j=1, NCtg} & (4) \\
 & \boldsymbol{V_{Mink} \le V_k^j \le V_{Maxk}} & \boldsymbol{\forall k=1, NBuses,\ \forall j=1, NCtg} & (5)
\end{array}$$

Where:

- $Margin_i^j$ – Stability margin for generator *i* under contingency j;
- Pg_i^j – Active power Pg for generator *i* under contingency *j*;
- Qg_i^j – Reactive power Qg for generator *i* under contingency *j*;
- V_k^j – Voltage at bus *k* for contingency *j*.

The objective function (1) corresponds to the maximization of the power system dynamic security index, which is associated with the stability margin. This margin is computed for each generator and for each contingency. The lowest margin among all generators and contingencies indicates how far from loosing stability is the power system. Restriction (2) is needed once a negative margin indicates that the system is unstable for, at least, one of the contingencies simulated. Restrictions (3) and (4) represent physical limits of equipments while restriction (5) corresponds to operative limits, like voltage profile. Active generations (Pg) are the controlled variables in the process of optimization.

A complete time domain simulation, that mimics the power system response to a contingency, is needed to compute the stability margin and the verification of the feasibility for each candidate solution. During the simulation, individualized transient energy functions [Jardim 94] compute potential and kinetic energies for each generator. Kinetic energy is transferred to the generator rotor during the fault. When the fault is cleared, this energy is converted back into potential energy, which is associated with the rotor angle shift with respect to the center of inertia of the system. Instability occurs when the maximum angle shift is reached and there is still kinetic energy left to be converted. This energy surplus is the negative margin of the generator. When the kinetic energy is fully converted into potential energy, the generator is stable and it is possible to estimate the amount of energy necessary to reach the maximum angle shift. This amount is the positive margin. Positive low margins indicate generators close to loose synchronism.

3 Evolutionary Algorithm LUDE

LUDE (*Line Up Differential Evolution*) algorithm was originally proposed to solve nonlinear constrained optimization problems [Sarimveis 05]. It is an arithmetic evolutive algorithm where the solutions of each generation are lined up according the fitness value, before the application of the crossover and mutation operators. The position in the list defines to what extent these operators affect the solution. Another important aspect is that there are only two parameters to be defined by the user, the population size and the number of iterations.

Like any other evolutionary algorithm, it is an iterative stochastic method that starts with a random population and build new generations by means of crossover and mutation operators. They are based on an improvement heuristic, where the worst solutions have bigger chances to suffer changes, while this probability is quite lower for the best solutions. The crossover operator corresponds to a linear combination of two consecutive solutions. If the solution is better then both parents, it assumes the position of the second parent (the worst). The same factor is used to combine all components of each solution.

In the mutation operator, each component of the solution may change, or not, according to a random choice. Besides, it randomly chooses any value in the valid range of that variable. The probability of changes is reduced from one solution to another, in the same generation and from one generation to another.

With respect to the crossover operator three modifications on the original algorithm were tested. The first is the use different factors (random) for each component of the solution, instead of using a single factor (uniform). The second variation is related with the choice of the pair of parents. The original algorithm uses two consecutive solutions. The modification proposed is to choose any solution, in a restricted list of candidates, which contains all solutions from the consecutive down to the last one. Finally, the last modification tested is related with the update rule. Originally a new solution is accepted only if it is better than both parents (selective). The modification accepts any new solution that is better than any of the two parents (greedy).

LUDE pseudo code (for minimization problem):

```
P0. Initialize population size L, MaxIt and make It=0;
P1. Build initial population: $x_i=x_i + r.(x_i^{up}-x_i^{lo})$,i=1,...,L,r =Rand(0,1);
P2. Make It = It + 1;
P3. Compute fit($x_i$),I=1,…,L;
P4. Sort: $x_i$ before $x_j$ if fit($x_i$)>fit($x_j$);
P5. Crossover:
        Do i=1, L-1
            $x_{iNEW} = x_i + r.(x_{i+1} – x_i)$, r =Rand(0,1)
            If fit($x_{iNew}$) < fit($x_{i+1}$) Then $x_i = x_{iNEW}$
        End Do
P6. Sort: $x_i$ before $x_j$ If fit($x_i$)>fit($x_j$);
P7. Mutation:
        Do i = 1, L
            $p_{mut} = (L – i + 1)/L$
            Do j=1,N
                r = Rand(0,1)
                If ($r < p_{mut}$) Then:
                    b = BynaryRand(0,1)
                    If b = 0 Then: $x_{iNew}(j) = x_i(j)+(x^{up}(j)-x_i(j)).(r/d).e^{(2.It/MaxIt)}$
                    If b = 1 Then: $x_{iNew}(j) = xi(j)+(xi(j)-xi^{lo}(j)).(r/d).e^{(2.It/MaxIt)}$
                End If
            End Do
            If fit($x_{iNEW}$) < fit($x_i$) Then: $x_i = x_{iNEW}$
        End Do
P8. Update Best
P9. If It = MaxIt Then: Stop; Else Go to P2.
```

Each array x_i corresponds to a different generation dispatch solution.

4 GRASP Algorithm

GRASP (*Greedy Randomized Adaptive Search Procedure*) is a hybrid construction and improvement metaheuristic suitable for combinatorial problems. Its principle is similar to the random multi-start method, where a local search is repeated for different random initial solutions. The basic difference is that the initial solutions are improved by the combination of random processes with heuristics, based on optimization premises.

GRASP pseudo code (for minimization problem):

```
P0. Define MaxIt and Seed;
P1. Do While k< MaxIt:
        a. Build a feasible solution: $\hat{x}$ (greedy and somehow random);
        b. Local Search for $\hat{x}$;
        c. If fit(neighbor($\hat{x}$)) < fit($\hat{x}$), Then: $\hat{x}$=neighbor($\hat{x}$) Go to P1.b;
        d. Update Best
        e. k=k+1;
P2. Return $\hat{x}$ and finish.
```

The initial solutions are generated based on the greedy premise to favor the dispatch of generators with bigger inertia, once, the bigger the total inertia, the more stable the system is. These solutions must define which generators to dispatch and the value of each generator dispatch.

The definition of two candidate lists makes the process of building an initial solution somehow random. The first one contains all generators sorted in a non crescent order of inertia values. The second list contains the following dispatch levels: 100%, 80%, 60%, 40%, 20%, and 0%. Each initial solution is built according to the following steps:

```
P0. Initialization:
        a. Estimate = Estimative of the total number of generators to be dispatched,
           considering load demand and maximum generation capacity of all generators;
        b. Compute Third = Estimate/3;
        c. Make NGen = 0.
P1. Select Generator:
        a. Choose one generator on the first list (those in the initial positions are more likely
           to be chosen);
        b. Remove the selected generator from the first list;
        c. Make NGen = NGen + 1.
P2. Define generation dispatch:
        a. Dispatch level is selected randomly on the second list, according to the following
           rules:
                i. If NGen < Third Then: Select dispatch in the range from 100% to 60%;
               ii. If Third < NGen < 2*Third Then: Select dispatch from 80% to 20%
              iii. If Third > 2*Third: Select dispatch from 60% to 0%.
P3. If (Total generation balances load) Then Stop: Else Go to P1.
```

The generation dispatch is defined in order to balance the load and losses. Therefore, some generators may not be dispatched at all.

The local search is based on a variation of the generation for each generator considering a positive and a negative variation around the initial value, whenever it was feasible. All combinations considering the selected generators are checked.

A "Path-Relinking" strategy was tested too. It considers the combination of each new solution with all solutions present in an elite set. This mechanism is applied when iterations reach 80% of the maximum number of iterations.

5 Results

Two different test systems were used. The first is the Anderson & Fouad test case, which has 9 buses and 3 generators. The second one is the IEEE14 with 14 buses and 6 generators.

Fitness function corresponds to the stability margin expressed in MW. It shows the estimative of the amount of MW that can be added to the generator dispatch with no risk of loosing stability.

5.1 Results for LUDE Algorithm

An initial evaluation of the influence of the maximum number of iterations and the population size was performed, considering the original formulation of the algorithm. Some variations, described in section 3, were tested too. The results are identified as follows:

- UF – Uniform Factor: Original formulation where a single factor is used in the combination of all components of the solution;
- RF – Random Factor: Modification where the combination factor changes randomly from one component to another;
- CS – Consecutive Solutions: Original formulation where crossover combine two consecutive solutions;
- RS – Random Solutions: Modification where one solution is combined with another selected randomly among its consecutive down to the last one;
- SU – Selective Update: Original formulation where a new solution is accepted only if it is better than both parents;
- GU – Greedy Update: Modification where a new solution is selected if it is better than any of its parents.

Each simulation was executed 5 times and the tables show the minimum, the maximum and the average values of best solution of all 5 simulations.

5.1.1 Anderson & Fouad Test Case

Table 5.1.1.1 shows the results for the variation of the maximum number of iterations, considering the LUDE original formulation and a population size of 10 solutions per generation.

Table 5.1.1.1 – Maximum number of iterations variation (Population=10)

UF-CS-SU	50 Iter	100 Iter	300 Iter	500 Iter	800 Iter	1000 Iter
Min	106,8	103,4	116,9	117,6	124,2	124,3
Max	185,2	123,0	124,1	125,1	127,1	127,0
Ave	126,5	112,6	121,8	123,3	126,1	126,2

A high value in the simulation with 50 iterations was verified. This caused an average value superior to the simulations with 300 and 500 iterations. This case is under investigation, but so far no special feature has been detected. If this value is purged, the average reduces to 113. The average value tends to stabilize around 800 iterations.

All further simulations will be done using 300 iterations, once the average is not too close to saturation.

Table 5.1.1.2 shows the results for the variation of population size, considering MaxIter=300.

Table 5.1.1.2 – Population size variation (300 Iterations)

UF-CS-SU	10 Solutions	30 Solutions	50 Solutions	100 Solutions
Min	116,9	123,5	124,6	123,6
Max	124,1	175,1	143,9	127,1
Ave	121,8	142,2	128,9	126,3

There were two inexplicable high results in the simulations with population size =30, resulting in a higher average comparing with other results. This case is under investigation, but so far no problem has been detected. If these values are purged, the average falls to 124.

Table 5.1.1.3 shows the results for the modifications in the crossover operator. These simulations were done considering maximum number of iterations =300 and population size = 10 solutions.

Table 5.1.1.3 – Crossover operator variations (300 Iterations e Population=10)

	UF-CS-SU	RF-CS-SU	UF-RS-SU	UF-CS-GU
Min	116,9	120,0	108,1	107,1
Max	124,1	124,7	125,7	126,7
Ave	121,8	122,3	117,3	118,3

These results show that the greedy update and the non consecutive (random) selection of parents increased the range of best solution values, but the average value is lower comparing to the original algorithm, showing that it did not improve the final results. The use of random factors showed promising results, with improvements in the minimum, maximum and average values, comparing to original uniform factor.

5.1.2 IEEE 14 Buses Test Case

Table 5.1.2.1 shows the results for the variation of the maximum number of iterations, considering the LUDE original formulation and a population size of 20 solutions per generation.

Table 5.1.2.1 – Maximum number of iterations variation (Population=20)

UF-CS-SU	300 Iter.	500 Iter	800 Iter	1000 Iter
Min	229,1	128,0	235,3	181,5
Max	234,5	235,0	236,4	236,2
Ave	232,1	200,6	235,8	224,3

In this case some low values were found for the simulations with 500 and 1000 iterations, resulting in very low averages. These cases are under investigation, but so far no hindrances have been detected.
All further simulations will be done using 300 iterations, once the average is not too close to saturation.
Table 5.1.2.2 shows the results for the variation of population size, considering MaxIter=300.

Table 5.1.2.2 – Population Size Variation (300 Iterations)

UF-CS-SU	20 Solutions	50 Solutions
Min	229,1	198,2
Max	234,5	235,7
Ave	232,1	219,8

Here again a low value was found for the simulations with 50 solutions, resulting in a lower average comparing to 20 solutions. This case is also under further investigation.

Table 5.1.2.3 shows the results for the modifications in the crossover operator. These simulations were done considering maximum number of iterations =300 and population size = 20 solutions.

Table 5.1.2.3 – Crossover operator variations (300 Iterations e Population = 20)

	UF-CS-SU	RF-CS-SU	UF-RS-SU	UF-CS-GU
Min	229,1	227,1	227,0	152,9
Max	234,5	235,1	234,7	235,4
Ave	232,1	232,1	231,8	201,2

These results show that the greedy update is not a good strategy for this case.

5.2 Results for GRASP Algorithm

The variables tested for this algorithm were the number of iterations, the generator restricted candidate list size, the MW variation value and the size of the elite set used in the “Path-Relinking”.

Each simulation was executed 5 times and the tables show the minimum, the maximum and the average values of best solution of all 5 simulations.

5.2.1 Anderson & Fouad Test Case

Due to its small size (3 generators), no parameterization was considered for the size of the generator restricted candidate list size. The parametric evaluation was done considering only the maximum number of iterations, the MW variation and the size of the elite set.

Table 5.2.1.1 shows the results for the MW variation, considering 10 iterations.

Table 5.2.1.1 – MW variation (10 Iterations)

	5MW	10MW	15MW	20MW	25MW	30MW	35MW	40MW	45MW	50MW
Min	49,2	81,9	63,1	75,2	84,5	84,5	97,1	103,2	58,1	103,2
Max	101,6	186,8	178,7	84,4	129,3	154,0	170,5	124,2	103,2	121,9
Ave	88,4	125,9	112,6	79,9	102,0	114,8	114,2	109,7	93,5	106,9

These results are not conclusive regarding the influence of bigger or smaller MW variations.

Table 5.2.1.1 shows the results for the MW variation, considering 30 iterations.

Table 5.2.1.2 – MW variation (30 Iterations)

	5MW	10MW	15MW	20MW	25MW	30MW	35MW	40MW	45MW	50MW
Min	72,3	74,6	79,2	58,1	79,8	66,7	97,2	103,2	103,2	103,2
Max	99,6	151,2	124,2	149,7	96,0	157,4	103,2	124,2	149,4	103,2
Ave	81,3	106,2	88,2	93,7	84,4	112,6	99,0	107,4	117,6	103,2

These results are not conclusive regarding the influence of bigger or smaller MW variations. The same happens comparing these results with those with 10 iterations.

Tabela 5.2.1.3 shows the results considering the variation of elite set size.

Table 5.2.1.3 – Elite set size variation (10 Iterations)

	10MW		30MW		50MW	
	Elite=3	Elite=10	Elite=3	Elite=10	Elite=3	Elite=10
Min	81,9	74,6	84,5	55,6	103,2	103,2
Max	186,8	158,9	154,0	103,2	121,9	103,2
Med	125,9	107,4	114,8	71,9	106,9	103,2

In this case both high and low values have been verified causing distortion on averages. This case deserves further investigation. “Path-Relinking”did not produce any improvement.

5.2.2 IEEE 14 Buses Test Case

The parametric evaluation considered the maximum number of iterations, the MW variation, the size of the generator restricted candidate list and the size of the elite set.

Table 5.2.2.1 shows the results for the MW variation, considering 10 iterations and a generator restricted candidate list with 3 generators.

Table 5.2.2.1 MW variation (10 Iterations and Gen. Restrict. Cand. List =3 generators)

	10MW	30MW	50MW
Min	157,1	183,3	170,8
Max	249,7	222,0	210,6
Ave	204,0	209,0	199,7

These results are not conclusive regarding the influence of bigger or smaller MW variations.

Table 5.2.2.1 shows the results for the MW variation, considering 10 iterations and a generator restricted candidate list with 6 generators.

Table 5.2.2.2 – MW variation (10 Iterations and Gen. Restrict. Cand. List =6 generators)

	10MW	30MW	50MW
Min	106,0	128,8	161,8
Max	261,5	226,4	220,2
Ave	170,2	186,7	194,2

The increase in the generator restricted candidate list ends up increasing the randomness of the initial solutions, which tend to be less optimized. This can be checked comparing these results with Table 5.2.2.1.

Table 5.2.2.3 shows the results considering the variation of elite set size

Table 5.2.2.3 – Elite set size variation (10 Iterat. and Gen. Restrict. Cand. List =3 generators)

	10MW	30MW	50MW
Min	113,5	180,1	161,8
Max	208,8	258,7	238,1
Med	173,2	205,3	204,7

“Path-Relinking”did not produce any improvement.

6 Conclusions

The methodology proposed in this work showed promising results. The use of complete time domain simulations in the assessment of fitness functions allows for the representation of more realistic scenarios. The performance of these algorithms is still an issue, but GRASP is much faster than LUDE.

LUDE and GRASP results are equivalents with a slight advantage for the first. The initial solution building process and local search heuristics, implemented in the GRASP algorithm, are quite simple but generated reasonable solutions. The local search cannot be applied for high dimension systems, once it considers all combinations of generation variations. This method deserves better strategies.

Total CPU time was extremely high. There has been an effort to identify simpler fitness functions that could be used, at least, in the first iterations, but without success. This area deserves further investigation.

The high and low values observed deserve further investigation too.

The variations on the crossover operator of LUDE algorithm were not encouraging. The greedy approach ended up with worse results.

7 References

[Edwards 96] Transient Stability Screening Using Artificial Neural Networks within a Dynamic Security Assessment System - Edwards A.R., Chan K.W., Dun R.W., Daniels A.R. - IEE Proceedings - Generation Transmission and Distribution, Vol.143 I.2., March 1996.

[Fouad 77] P.M.Anderson and A.A. Fouad, Power System Control and Stability, Vo. 1, The Iowa State University Press, Ames, Iowa, USA, 1977;

[Groom 96] Real-Time Security Assessment of Electrical Power Systems- Groom C.G., Chan K.W., Dunn R.W., Daniels A.R. - IEEE Transactions on Power Systems, Vol.11, I.2, May 1996.

[Jardim 94] Advances in Power System Transient Stability Assessment Using Transient Energy Function Methods - Jardim J.L.A. - Phd Thesis at University of London, 1994.

[Kamwa 01] Time-Varying Contingency Screening for Dynamic Security Assessment Using Intelligent-Systems Techniques - Kamwa I., Grondin R., Loud L. - IEEE Transactions on Power Systems, Vol. 16, I.3, August 2001.

[Kuo 95] A Generation Rescheduling Method to Increase The Dyamic Security of Power Systems- Kuo D. e Bose A. - IEEE Transactions on Power Systems, Vol. 10, No. 1, February 1995.

[Pao 92] Combined Use of Unsupervised and Supervised Learning for Dynamic Security Assessment - Pao Y.H., Sobajic D.J. - IEEE Transactions on Power Systems, Vol.7, I.2, May 1992.

[Sarimveis 05] A line up evolutionary algorithm for solving nonlinear constrained optimization problems – Sarimveis H. e Nikolakopoulos A. – Computers & Operations Research – Vol.32, I.6, pp.1499-1514; June 2005.

[Sobajic 89]	Artificial Neural-Net Based Dynamic Security Assessment for Electric Power Systems - Sobajic D.J., Pao Y.H. - IEEE Transactions on Power Systems, Vol.4, I.1, February 1989.
[Sun 04]	Approach for Optimal Power Flow with Transient Stability Constraints- Sun Y., Xinlin Y. e Wang H. F. - IEE Proc. Gener. Transm. Distrib., Vol. 151, No. 1, January 2004.
[Tso 95]	Fuzzy-Set Approach to Dynamic Voltage Security Assessment - Tso S.K., Zhu T.X., Zeng Q.Y. e Lo K.L. - IEE Proceedings - Generation, Transmission and Distribution, March 1995.
[Yuan 03]	A Solution of Optimal Power Flow with Multicontingency Transient Stability Constraints - Yuan Y., Kubokawa J. e Sasaki H. - IEEE Transactions on Power Systems, Vol. 18, No. 3, August 2003