# Solving the Point of Collapse Problem using a Heterogeneous Computer Network

**Benjamín Barán, Freddy Cardozo, José Atlasovich, Christian Schaerer**
**Centro Nacional de Computación**
**Universidad Nacional de Asunción**
**P.O.Box: 1439 - Campus Universitario**
**San Lorenzo, Paraguay**

## ABSTRACT

This paper presents a parallel method, based on the sequential *Point of Collapse* method (PoC), to calculate the steady state voltage stability limit (point of collapse or critical point) of potentially large electrical power systems, using a distributed (and possible asynchronous) computer environment composed of a network of workstations or PCs. The parallelization of the sequential PoC method is based on a recently formalized technique called *Team Algorithm* (TA). Experimental results are presented, proving the advantages of the proposed method using a network of computers instead of the traditional single processor computer.

**Keywords**: Voltage Stability, Distributed Computer Environment, Critical Point.

## 1. INTRODUCTION

As electrical systems started to be demanded to operate in a way to achieve their maximum transmission capability, they started to suffer from voltage stability problems and in some cases they even collapsed. Therefore, the research of the Point of Collapse problem has been receiving great attention in the last few years [1]. One research approach is to associate the collapse phenomenon with the maximum load the system can withstand. Based on this proposal, some tools were developed to calculate such point under certain assumptions [2,3]. The present work follows the above approach.

With respect to distributed processing using a network of workstations or PCs, important progress has been made concerning the development of low-cost technologies with high processing capacity. At the present time, there is a world-wide tendency to use interconnected computers via networks [4]. However, one of the major drawbacks is the lack of adequate software to exploit its potential advantages in speedup, low cost to performance ratio, scalability, flexibility, etc.

This paper is based on two different areas of research: the Point of Collapse Problem in an electrical power system and the parallel and distributed processing using Team Algorithms (TA). The latter is a computational technique that allows solving a system of equations using hybrids of different methods, taking advantage of the parallelism in a distributed computer system, even in the presence of asynchronism [5,6].

The main idea behind a Team Algorithm is to partition a large-complex problem in several subproblems, solving each part in a different processor of the computer network. To do so, each processor may use a different method that updates its assigned variables, transmitting the updated values and receiving the variables updated by other processors, without blocking or interruptions. The calculation proceeds iteratively until global convergence is achieved. A general convergence analysis considering an asynchronous environment is studied in [6].

Some of the most outstanding features of the TA are: to achieve a solution of algebraic equations that each of the combined algorithms can not solve individually, and to obtain a significant speedup when large scale problems are divided in smaller subproblems that can be easily solved in parallel.

The major contribution of this work is to present a parallel method that efficiently calculates the point of collapse using a workstation or PC network that already exists in most organizations, instead of using expensive uniprocessor systems that are not always easily available in small organizations.

Section 2 presents the point of collapse problem. The point of collapse method (PoC) and the proposed parallelization schemes are presented in Section 3. Experimental results are presented in Section 4, leaving the conclusions for Section 5.

## 2. THE POINT OF COLLAPSE PROBLEM

The system of equations that represents the steady state behavior of a power system has the form [7]:

$$F(\mathbf{x},\mathbf{\Pi}) = \begin{bmatrix} f_1(\mathbf{x},\mathbf{\Pi}) \\ \vdots \\ f_N(\mathbf{x},\mathbf{\Pi}) \end{bmatrix} = \mathbf{0} \qquad (1)$$

where $F(\cdot,\cdot)$: $\mathrm{R}^N \times \mathrm{R}^M \mapsto \mathrm{R}^N$,
$\mathbf{x},\mathbf{0} \in \mathrm{R}^N$ and $\mathbf{\Pi} \in \mathrm{R}^M$.

Here, $\mathbf{x}$ is the power system state vector (electrical bus voltage) and $\mathbf{\Pi}$ is a parameter vector of real and reactive load powers. Although several valid operating states $\mathbf{x}$ are associated to a certain distribution $\mathbf{\Pi}^o$ (under normal operating conditions), just one of them is of operative interest, which is denoted by $\mathbf{x}^o$.

As the load vector $\mathbf{\Pi}$ varies from the current operating load $\mathbf{\Pi}^o$, its corresponding steady state also varies according to (1). Upon reaching a certain load distribution $\mathbf{\Pi}^*$ at the busbars, equation (1) reaches a unique steady state solution $\mathbf{x}^*$, hence $(\mathbf{x}^*,\mathbf{\Pi}^*)$ is denominated *point of collapse*. This point has the property that for a larger load at the busbars there is no steady state solution. In addition, the Jacobian matrix $F_x$ of (1) is singular at that point, having a unique zero eigenvalue with non-zero left and right eigenvectors [2].

Different load growth directions can be established at the operating point $\mathbf{\Pi}^o$, determined by the unitary vector $\mathbf{B}$ in the $\mathrm{R}^M$ space; therefore, the load evolution could be represented by

$$\mathbf{\Pi} = \mathbf{\Pi}^o + \mathbf{B}\lambda \qquad (2)$$

where $\lambda$ is named *scalar load parameter* [3].

For each load growth direction there is a possible point of collapse. The set of these points in $\mathrm{R}^M$ is a hyper-surface, namely $\Sigma$, which limits the operating steady state space of the system. Thus, for a point which belongs to $\Sigma$, the following equality holds:

$$\mathbf{\Pi}^* = \mathbf{\Pi}^o + \mathbf{B}\lambda^* \qquad (3)$$

The line segment between $\mathbf{\Pi}^o$ and $\mathbf{\Pi}^*$ is the set of possible load vectors in $\mathbf{B}$'s direction [7] and $\lambda^*$ represents a system stability index which determines the proximity of the point of collapse to the operating condition in the specified direction.

## 3. THE POINT OF COLLAPSE METHOD

As the loads are increased and the electrical system approaches its load capability limit, the convergence of the load flow solution becomes more and more difficult using traditional methods such as the Newton-Raphson (NR). This is due to the aforementioned characteristic of system (1)'s Jacobian singularity at the critical point. One of the techniques used to overcome this problem is the Point of Collapse Method, which consists of solving the following system of equations [2]:

$$\Phi = \begin{bmatrix} F(\mathbf{x},\mathbf{\Pi}) \\ F_{\mathbf{x}}^T(\mathbf{x},\mathbf{\Pi})\mathbf{w} \\ \mathbf{c}^T\mathbf{w} - 1 \end{bmatrix} = \mathbf{0} \qquad (4)$$

where $\mathbf{c} \in \mathrm{R}^N$ is a constant non-zero vector. On one hand, $F(\mathbf{x},\mathbf{\Pi}) = \mathbf{0}$ guarantees that $\mathbf{x}$ is a solution of the load flow; on the other hand, the equation $F_{\mathbf{x}}^T(\mathbf{x},\mathbf{\Pi})\mathbf{w} = \mathbf{0}$ with $\mathbf{c}^T\mathbf{w} - 1 = 0$, ensures that $\mathbf{x}$ and the left eigenvector $\mathbf{w}$ of $F_x$ correspond to the Point of Collapse with an eigenvalue zero and a non-zero eigenvector.

If we consider the parameter load vector as stated in (2), system (4) can be rewritten as:

$$\Phi = \begin{bmatrix} F(\mathbf{x},\lambda) \\ F_{\mathbf{x}}^{T}(\mathbf{x},\lambda)\mathbf{w} \\ \mathbf{c}^{T}\mathbf{w}-1 \end{bmatrix} = \mathbf{0} \qquad (5)$$

which is a nonlinear system of equations of dimension $n = 2N+1$.

To solve (5) using a distributed computing system, a good partitioning of the electrical system is needed. That may be obtained using any published method [8,9] that profits from the natural decoupling of the system. Considering a system consisting of *p* processors, the resulting partition of $F(\mathbf{x}) = \mathbf{0}$ may be written as:

$$F(\mathbf{x}) = \begin{bmatrix} F_1(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_p) \\ F_2(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_p) \\ \vdots \\ F_p(\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_p) \end{bmatrix} = \mathbf{0} \qquad (6)$$

where $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \cdots & \mathbf{x}_p^T \end{bmatrix}^T$ is the power system state vector. For a good global convergence in a distributed computer system, each function $F_i(\mathbf{x}_1,\cdots,\mathbf{x}_i,\cdots,\mathbf{x}_p)$ should predominantly depend on $\mathbf{x}_i$ (no dependence on $\mathbf{x}_k \ \forall k \neq i$ implies perfect decoupling) [6].

Since $F(\mathbf{x},\lambda) = \mathbf{0}$ is the load flow equation including the scalar load parameter $\lambda$, we can use the partition criteria mentioned above. Hence, equation (5) can be rewritten as:

$$\Phi = \begin{bmatrix} F_1 \\ \vdots \\ F_p \\ \sum_{k=1}^{p} F_{k\mathbf{x}_1}^{T}\mathbf{w}_k \\ \vdots \\ \sum_{k=1}^{p} F_{k\mathbf{x}_p}^{T}\mathbf{w}_k \\ \sum_{k=1}^{p} \mathbf{c}_k^{T}\mathbf{w}_k - 1 \end{bmatrix} = \mathbf{0} \qquad (7)$$

where $\mathbf{w} = \begin{bmatrix} \mathbf{w}_1^T & \mathbf{w}_2^T & \cdots & \mathbf{w}_p^T \end{bmatrix}^T$.

According to the above criteria, our partition is such that $F_{k\mathbf{x}_i}^{T} \ \forall k \neq i$ is relatively small compared to $F_{i\mathbf{x}_i}^{T}$. Therefore, equations $\sum_{k=1}^{p} F_{k\mathbf{x}_i}^{T}\mathbf{w}_k$ depend mostly on $(\mathbf{x}_i,\mathbf{w}_i)$. It follows that a good assignment for processor *i* would be:

$$\Phi_i = \begin{bmatrix} F_i \\ \sum_{k=1}^{p} F_{k\mathbf{x}_i}^{T}\mathbf{w}_k \end{bmatrix} = \mathbf{0} \qquad (8)$$

On the other hand, the last equation of (7) depends on all $\mathbf{w}_i$; making it difficult to decide *a priori* where it will best be solved. Three possible solution schemes follow.

**First Scheme**

The first scheme consists of assigning the last equation of (7) to one processor, for example:

$$\Phi_1 = \begin{bmatrix} F_1 \\ \sum_{k=1}^{p} F_{k\mathbf{x}_1}^{T}\mathbf{w}_k \\ \sum_{k=1}^{p} \mathbf{c}_k^{T}\mathbf{w}_k - 1 \end{bmatrix} = \mathbf{0}$$

$$\Phi_i = \begin{bmatrix} F_i \\ \sum_{k=1}^{p} F_{k\mathbf{x}_i}^{T}\mathbf{w}_k \end{bmatrix} = \mathbf{0} \quad \forall i \neq 1 \qquad (9)$$

where processor 1 solves $\Phi_1 = \mathbf{0}$ updating $(\mathbf{x}_1, \mathbf{w}_1, \lambda)$; while a processor *i* *(i ≠ 1)* solves $\Phi_i = \mathbf{0}$ updating $(\mathbf{x}_i, \mathbf{w}_i)$. Convergence by this scheme was not achieved in any of the tested problems.

**Second Scheme**

In order to achieve convergence, it was decided to assign the last equation of (7) to all processors, such that:

$$\Phi_i = \begin{bmatrix} F_i \\ \sum_{k=1}^{p} F_{k\mathbf{x}_i}^{T} \mathbf{w}_k \\ \sum_{k=1}^{p} \mathbf{c}_k^{T} \mathbf{w}_k - 1 \end{bmatrix} = \mathbf{0} \quad \forall i \qquad (10)$$

The implementation of this procedure, known as *partial overlapping* [10], is illustrated in Figure 1, where an equation with weights $\omega_k$ and $\omega_\lambda$ selected such that

$$\omega_\lambda + \sum_{k=1}^{p} \omega_k = 1 \qquad (11)$$

is used to make sure that all $\lambda_k$ converge to the same solution $\lambda^*$ [5].

In Figure 1, $G_i$ and $G_j$ are the algorithms that update the variables, in this case, the NR's method, of processors *i* and *j* respectively.

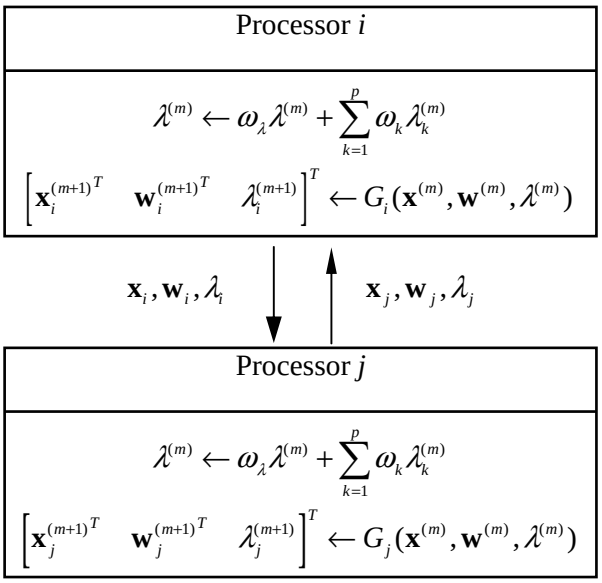


**Figure 1.** Second and third schemes

Particularly, it can be shown that for $p = 2$, at any iteration *m*, the following equality holds:

$$\begin{aligned} \mathbf{c}_1^{T} \mathbf{w}_1^{(m+2)} &= \mathbf{c}_1^{T} \mathbf{w}_1^{(m)} = \text{const.} \\ \mathbf{c}_2^{T} \mathbf{w}_2^{(m+2)} &= \mathbf{c}_2^{T} \mathbf{w}_2^{(m)} = \text{const.} \end{aligned} \qquad (12)$$

This imposes an additional restriction on the possible directions of eigenvector **w** with respect to the original equation (5). In fact, convergence could not be achieved in any of the tested problems indicated in Tables 1 and 2 using this scheme. Therefore a third scheme, also using partial overlapping, is proposed.

**Third Scheme**

To overcome the above problem, it is possible to modify the last equation of (10) by:

$$\mathbf{c}^T \mathbf{w} + \kappa\lambda - \xi = 0 \qquad (13)$$

where $\kappa$ and $\xi$ are constants. Equation (13) still guarantees a non-zero eigenvector **w** without the additional restriction previously mentioned. Introducing this modification into equation (10) yields to:

$$\Phi_i = \begin{bmatrix} F_i \\ \sum_{k=1}^{p} F_{k\mathbf{x}_i}^{T} \mathbf{w}_k \\ \sum_{k=1}^{p} \mathbf{c}_i^{T} \mathbf{w}_i + \kappa\lambda_i - \xi \end{bmatrix} = \mathbf{0} \qquad (14)$$

This third scheme can still be represented by the diagram in Figure 1 and it gives good experimental results, as shown in Tables 1 and 2.

## 4. EXPERIMENTAL RESULTS

The described schemes were implemented in a heterogeneous network of workstations (a SUN SPARC-Station 5 & a DEC 3000) and several Personal Computers (PCs) using PVM (Parallel Virtual Machine) libraries in an extended version of ANSI C. Proposed schemes were tested in three different implementations: sequential, parallel synchronous and parallel asynchronous versions. However, for reasons of shortness and for establishing an easier comparison frame between sequential and parallel algorithms, the experimental results presented in this section are limited to a distributed system of a PCs with Intel Pentium processors of 100 MHz and 8 MB of RAM, running under LINUX operating system.

Experimental results are presented in Tables 1 and 2 for IEEE's 30 and 118 busbar systems, respectively. These results were obtained for $\omega_\lambda = 0$, $\kappa = 3$, $\xi = 10$, $\mathbf{c} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$ and for a tolerance $\varepsilon = 0.05$.

The *speedup* is defined as:

$$S_p = \frac{\text{best sequential time}}{\text{parallel method time}} \tag{15}$$

As for the initial condition, the same criteria used in the traditional sequential solution method [2] was used for IEEE 30 busbar system. However, in the second test problem, the IEEE 118 busbar system, the initial condition together with the partition method proved to be critical for convergence. Initial estimates were selected close to the solution with a deviation defined by:

$$\Delta\mathbf{z}^{(0)} := \left[ \left(\mathbf{x}^* - \mathbf{x}^{(0)}\right)^T \quad \left(\mathbf{w}^* - \mathbf{w}^{(0)}\right)^T \quad \left(\lambda^* - \lambda^{(0)}\right) \right]^T \tag{16}$$

As for the partition criteria, the method presented in [11] was used in all of the tested problems.

The major implementations performed were:

**Sequential**

Traditional solution of equation (5) using a single processor.

**Synchronous TA**

Each processor solves its corresponding set of equations (14), exchanging the updated values after each iteration (see Figure 1); i.e. an administrator program waits until all processors have finished their iteration and assembles the new set of variables broadcasting it to all processors so they can perform another iteration until all errors are within tolerance.

A simplified algorithm for a generic processor *i* follows:

```
do while (no global convergence)
   wait until complete vector (x,w,λ) is
   received;
   check local convergence;
   if local convergence is not achieved
   update (x_i,w_i,λ_i) solving Φ_i = 0 by NR;
   send (x_i,w_i,λ_i) to the administrator;
```

And for the administrator:

```
do while (no global convergence)
   wait until all (x_k,w_k,λ_k) are received;
   broadcast complete updated vector
   (x,w,λ) to all processors;
   if all processors detect local
   convergence, global convergence is
   achieved;
```

**Asynchronous TA**

Each processor solves its corresponding set of equations (14), exchanging the updated values after each iteration, as in the method previously described, but without blocking or interruptions. Before starting a new iteration, the most recently updated values of the variables received so far are read and after finishing the iteration, each processor broadcasts its updated variables to the other processors. If all processors detect their local errors within tolerance, the administrator assembles a complete set of variables and broadcasts it to all processors so they can perform a convergence check with the same global possible solution vector.

A simplified algorithm for a generic processor *i* follows:

```
do while (no global convergence)
   if the administrator sends a complete
   vector (x,w,λ), check for local
   convergence;
   if new information is received from
   processor k, update (x_k,w_k,λ_k);
   check local convergence and update
   (x_i,w_i,λ_i) solving Φ_i = 0 by NR;
   broadcast (x_i,w_i,λ_i) to all processors
   k ≠ i;
   if local convergence was achieved, also
   send (x_i,w_i,λ_i) to the administrator;
```

For the administrator:

```
do while (no global convergence)
   wait until all (x_k, w_k, λ_k) are received;
   broadcast complete vector (x, w, λ) to
   all processors;
   wait until all processors have checked
   for local convergence.  Under these
   circumstances, if all processors detect
   local convergence again, global
   convergence is achieved;
```

**Table 1.** Experimental results for IEEE 30

| Number of Processors | Number of Iterations | Time [s] | Speedup $S_p$ |
|---|---|---|---|
| 1 | 7 | 11 | 1.00 |
| 2 | 20 | 7 | 1.57 |
| 3 | 42 | 5 | 2.20 |
| 4 | 41 | 4 | 2.75 |

**Table 2.** Experimental results for IEEE 118

| Number of Processors | Method | Initial Condition's deviation $\|\Delta \mathbf{z}^{(0)}\|$ [%] | Number of Iterations | Time [s] | Speedup $S_p$ |
|---|---|---|---|---|---|
| 1 | seq. | 1 - 5 | 3 | 161 | 1.00 |
| 2 | S | 3 | 5 | 90 | 1.79 |
| | A | 3 | -- | 79 | 2.04 |
| 3 | S | 3 | 23 | 70 | 2.30 |
| | A | 3 | -- | 19 | 8.47 |
| 4 | S | 3 | 19 | 31 | 5.19 |
| | A | 3 | -- | 38 | 4.24 |
| 5 | S | 3 | 17 | 13 | 12.38 |
| | A | 3 | -- | 16 | 10.06 |

Note: seq. Sequential
S Parallel, synchronous TA
A Parallel, asynchronous TA

## 5. CONCLUSIONS

The present paper relates two different areas of research: the Point of Collapse Problem in an electrical power system, and the parallel and distributed processing using Team Algorithms [12].

After presenting three different implementation schemes for parallelizing the PoC method, it is concluded that the best scheme uses partial overlapping and reformulates the original equations.

By considering the experimental results presented in Tables 1 and 2, it was possible to conclude that:

- The advantage of partial overlapping is demonstrated when there exist critical equations that can not be assigned to only one processor.
- In all of the tested problems where convergence was achieved, the use of parallelism enables substantial reduction in the execution time compared to the sequential method, even in a small problem such as the IEEE 30 (Table 1).
- The major limitation of the proposed method is the partitioning of the system. In fact, the partitioned system has a higher sensibility to initial conditions and it is, in general, much harder to converge.
- Considering the IEEE 118 system, it can be seen in Table 2 that the asynchronous TA converged faster than the synchronous for two and three processors. Particularly, for $p = 3$ Table 2 shows an outstanding asynchronous speedup due to a synergetic effect [6]. For $p = 4$ and 5, however, the asynchronous TA reached the solution slower than the synchronous but still with a high speedup.

In short, this paper presents an efficient method that enables the solution of the Point of Collapse Problem using an existing network of PC's or workstations instead of the more traditional but expensive uniprocessor systems.

## 6. ACKNOWLEDGMENTS

The authors would like to express their gratitude to Rodrigo Ramos, Diana Benítez and Edgar Sánchez for their valuable support.

## 7. REFERENCES

[1] P. Kundur and B. Gao, "Practical Considerations in Voltage Stability Assessment", *IV Symposium of specialists in Electric Operational and Expansion Planning (IV SEPOPE)*, Foz do Iguaçu - Brazil, May 1994.

[2] C. Cañizares and F. Alvarado, "Point of Collapse and Continuation Methods for Large AC/DC Systems", *IEEE Trans. on Power System*, Vol. 8, No. 1, February 1993.

[3] V. Ajjarapu and C. Christy, "The Continuation Power Flow: A Tool for Steady State Voltage Stability Analysis", *IEEE Trans. on Power Systems*, Vol. 7, No. 1, pp. 416-423. February 1992.

[4] F. Wu and L. Murphy, "Parallel and Distributed Processing: Applications to Power Systems", *IV Symposium of Specialists in Electric Operational and Expansion Planning (IV SEPOPE)*, Foz do Iguaçu - Brazil, May 1994.

[5] B. Barán, E. Kaszkurewicz, and D. M. Falcão, "Team Algorithms in Distributed Load Flow Computations", *IEE Proc. Gener. Transm. Distrib.*, Vol. 142, No. 6, pp. 583-588, November 1995.

[6] B. Barán, "*Estudo de Algoritmos Combinados Paralelos Assíncronos*", Doctoral Dissertation, COPPE/UFRJ, Rio de Janeiro - Brazil, October 1993.

[7] I. Dobson and L. Lu, "New Methods for Computing a Closest Saddle Node Bifurcation and Worst Case Load Power Margin for Voltage Collapse", *IEEE Trans. on Power Systems*, Vol. 8, No. 3, pp. 905-913, August 1993

[8] M. Vale, D.M. Falcao and E. Kaszkurewicz, "Electrical Power Network Decomposition for Parallel Computations", *IEEE International Symposium on Circuits and Systems (ISCAS 92)*, San Diego - California, 1992.

[9] A. El-Keib, J. Neiplocha, H. Sing and D. J. Maratukulam, "A Decomposed State Estimation Technique Suitable for Parallel Processor Implementation", I*EEE Trans. on Power Systems*, Vol. 7, No. 3, August 1993.

[10] M. Ikeda and D. Siljak, "Overlapping Decomposition, Expansions and Contractions of Dynamic Systems", *Large Scale System 1*, North-Holland Publishing Co., pp. 29-38, 1980.

[11] B. Barán, D. Benítez and R. Ramos, "Partición de Sistemas de Ecuaciones para su Resolución Distribuida", *XXII Conferencia Latino - Americana de Informática* (PANEL 96), Bogotá - Colombia, June 1996.

[12] B. Barán, E. Kaszkurewicz and A. Bhaya, "Distributed Asynchronous Team Algorithms: Application to the Load Flow Problem", *XIX Conferencia Latino - Americana de Informática*, Buenos Aires - Argentina, September 1993.