

TEAM ALGORITHMS

Uma Aplicação para Fluxo de Potência Elétrica

Djalma M. Falcão ¹ e Benjamín Barán ²
Universidade Federal do Rio de Janeiro
Caixa Postal 68516 - CEP 21945
Rio de Janeiro - RJ - Brasil

Resumo

O trabalho apresenta os *Algoritmos Combinados*, mais conhecidos na literatura como *Team Algorithms* [12] estudando sua potencialidade no contexto da computação paralela. São apresentados exemplos demonstrando que o *Team Algorithm* consegue resolver problemas que os algoritmos que estão sendo combinados não conseguem resolver individualmente.

Um *Team Algorithm* é utilizado para resolver o problema de *Fluxo de Potência Elétrica* combinando 2 algoritmos com características bem diferenciadas: o Desacoplado Rápido, derivado do Newton-Raphson, e uma variante do Método da matriz Y, usando o algoritmo de Jacobi.

Diferentes políticas na escolha dos pesos do algoritmo são discutidas e resultados experimentais são apresentados para 9 problemas tipos (Sistemas IEEE).

Palavras Chaves:

Team algorithms, paralelismo, fluxo de potência, algoritmos de Jacobi e Desacoplado Rápido.

1 Introdução

A primeira proposta para utilizar *Algoritmos Combinados* na solução do Fluxo de Potência aparece em 1971 [5] tentando combinar as excelentes características de convergência do algoritmo de Newton-Raphson, com a menor complexidade do Método de Jacobi. Os resultados apresentados no trabalho não eram desalentadores, mas na prática não chegaram a ter muita repercussão devido à tecnologia computacional da época, puramente seqüencial, para a qual resultava um desperdício de memória ter varios algoritmos trabalhando concorrentemente.

Com a aparição dos computadores paralelos na década de '80, muitas idéias são revisadas. Em 1983 aparece um trabalho que postula novamente a utilização de algoritmos

¹Prof. Adjunto da Univesidade Federal do Rio de Janeiro - Brasil.

²Prof. Adjunto da "Univesidad Nacional de Asunción - Paraguay".

combinados [12]; sendo que desta vez, para computadores paralelos de forma que cada processador trabalhe com um algoritmo só, eliminando assim o problema de desperdício de memória da proposta seqüencial. Esse trabalho denomina a esta combinação de algoritmos como *Team Algorithm*. Passada quase uma década, a tecnologia da computação paralela está suficientemente amadurecida para tirar proveito da utilização dos *Team Algorithms* [8, 9]. O objetivo do presente trabalho é estudar a potencialidade das idéias de [12] e sua utilização na solução de problemas de Fluxo de Potência Elétrica.

O trabalho é dividido em 5 seções. A seção 2 trata das idéias gerais do *Algoritmo Combinado*. A seção 3 apresenta o problema do Fluxo de Potência Elétrica, deixando as alternativas mais complexas de solução para a seção 4. Finalmente, a seção 5 apresenta as conclusões do trabalho.

2 Team Algorithm. O problema.

Muitos dos problemas computacionais conhecidos podem ser resolvidos por métodos bem diversos, cada qual com suas vantagens próprias. A escolha do melhor método nem sempre é simples, pois qualquer que seja a escolha, algum outro método pode ter outras características desejadas. Desta forma, a idéia de combinar algoritmos para tentar obter as melhores características de cada método deu origem aos *Team Algorithms*. O primeiro trabalho neste sentido [12] propõe resolver

$$F(X) = \vec{0} \quad (1)$$

onde: $X \in \mathbb{R}^n$, e $F: \mathbb{R}^n \rightarrow \mathbb{R}^n$, combinando as excelentes qualidades de convergência do algoritmo de Newton-Raphson com uma variante do Método do Gradiente.

A idéia é combinar os algoritmos usando pesos $\alpha(k)$ e $\beta(k)$ conforme à equação:

$$X(k+1) = X(k) + \alpha(k) \Delta X_{NR}(k) + \beta(k) \Delta X_G(k) \quad (2)$$

onde: $k = 0, 1, 2, \dots$ representa o número da iteração e os $\Delta X(k)$ são as variações no vetor X calculadas por cada algoritmo na iteração k .

Em particular, para a escolha dos algoritmos acima mencionados:

$$\Delta X_{NR}(k) = -[J(k)]^{-1} F(X(k)) \quad (3)$$

$$\Delta X_G(k) = -\gamma [J(k)]^T F(X(k)) \quad (4)$$

onde: $J(k) = \left\{ \frac{\partial F_i}{\partial X_j} \right\}$ é o Jacobiano calculado na iteração k e γ é um parâmetro de relaxação ($0 < \gamma < 1$) escolhido suficientemente pequeno de forma a assegurar a convergência.

A versão síncrona do algoritmo 2 é bem simples: um processo *Administrador* envia o valor da iteração em curso $X(k)$ a cada algoritmo para que estes calculem os ΔX em paralelo e combina os resultados escolhendo os pesos adequados. O algoritmo continua

iterando até que o resíduo $E(X) := \|F(X)\|$ seja menor que um dado Erro Aceitável ε . A título de ilustração, seja

$$F(X) = \begin{bmatrix} F_1(X) \\ F_2(X) \\ F_3(X) \\ F_4(X) \end{bmatrix} = \begin{bmatrix} x_1 + 10x_2 \\ \sqrt{5}(x_3 - x_4) \\ (x_2 - 2x_3)^2 \\ \sqrt{10}(x_1 - x_4)^2 \end{bmatrix}, \quad X \in \mathbb{R}^4 \quad (5)$$

Inicialmente analisemos uma escolha a priori dos pesos sem variação ao longo da execução. Duas escolhas dos pesos são especialmente interessantes.

($\alpha = 1$; $\beta = 0$): que resulta no primer algoritmo, o Newton-Raphson,

($\alpha = 0$; $\beta = 1$): que resulta no algoritmo do Gradiente.

Consideremos os resultados experimentais de um caso típico usando o resíduo $\|F(X)\|_2$ e $\varepsilon = 0.01$.

Exemplo 1: $X(0) = [1 \ 0 \ 0 \ 1]^T$

Para esta condição inicial, o algoritmo de Newton-Raphson não pode ser utilizado, pois o Jacobiano é singular. Por sua parte, o algoritmo do Gradiente não consegue convergir em menos de 200 iterações³. A alternativa de utilizar um *Team Algorithm* que permita sair da região onde o Jacobiano é singular e então aproveitar as excelentes características do Newton-Raphson, torna-se evidente. De fato, para $\alpha = \beta = 0.8$ o *Algoritmo Combinado* converge em 3 iterações.

O exemplo mostra claramente as vantagens do *Team Algorithm*. Para esclarecer o comportamento do algoritmo, pesquisamos o número I de iterações necessárias até a convergência, para diferentes pesos. Os valores da função:

$$I = f(\alpha; \beta) > 0 \quad (6)$$

podem ser apresentados de forma gráfica no 'plano $\alpha - \beta$ ', como ilustra o mapa da Figura 1.

Na Figura 1 pode-se notar uma *Região de Convergência Ótima* na qual se tem a melhor convergência possível ($I = 3$). A medida que o ponto de trabalho se afasta desta região, o número de iterações I cresce.

Variando a condição inicial é possível obter outros mapas similares [1]. Uma característica encontrada repetidas vezes em diversos problemas é que a *Região de Convergência Ótima* inclui alguns pontos da reta $\beta = 0$ que em geral ocorrem para $\alpha > 1$. Isto sugere que o algoritmo de Newton-Raphson pode ser acelerado usando um *Parâmetro de Sobre-Relaxação*⁴ [2]. Como exemplo, isto sucede com $X(0) = [3 \ -1 \ 0 \ 1]^T$.

³O algoritmo do Gradiente requer 266 iterações.

⁴Idéia similar à utilizada nos algoritmos SOR (Successive Over-Relaxation).

β								
1.2	221	11						
1.0	266	10	6		*	*		
0.8	333	11	6	4	3	3		
0.6		11	6	4	3	5	33	
0.4			7	4	3	4	6	
0.2				5	3	3	4	
0	*	*	*	*	*	*	*	
	0	0.2	0.4	0.6	0.8	1.0	1.2	α

Figura 1: Mapa do Exemplo 1 ilustrando como varia o número de iterações I no plano definido pelos pesos α e β . O * indica que o algoritmo não converge para esse ponto.

Outros detalhes sobre as características dos *mapas* são discutidos em [1] onde são apresentados exemplos para os quais o número de iterações I é mínimo para valores negativos de β e $\alpha > 1$. Em outras palavras, a combinação ótima utiliza o Gradiente para divergir ligeiramente, de forma a permitir uma melhor convergência do Newton-Raphson. Uma situação que nos faz lembrar idéias do 'Simulated Annealing' [4].

Pode-se pesquisar o que acontece se o *Administrador* tem 'inteligência' suficiente para escolher os melhores pesos a cada iteração, segundo um dado critério. Desta forma, os *Pesos* $\alpha(k)$ e $\beta(k)$ da equação 2 são função da iteração k . Os métodos para a escolha 'inteligente' dos *pesos* podem ser vários. Para o exemplo 1 foram testadas varias minimizações do resíduo E , em diferentes normas. A que apresentou o melhor desempenho em todos os casos foi a que minimiza $\|F(X)\|_\infty$. Assim, no exemplo 1 se obteve o menor resíduo depois de 3 iterações.

Postulando um *Administrador inteligente*, utilizamos uma definição de erro E_r que utiliza a solução X_0 na etapa de *aquisição de dados* com o objetivo de descobrir a melhor escolha dos *Pesos* de forma a minimizar um *Erro Linear* E_r definido como:

$$E_r = \|X(k+1) - X_0\|_p = \|X(k) + \alpha(k) \Delta X_{NR}(k) + \beta(k) X_G(k) - X_0\|_p \quad (7)$$

A minimização utilizando a equação 7 para $p = 2$ é conhecida na literatura [11] e se limita ao cálculo da *Pseudoinversa* $A^+ := (A^T A)^{-1} A^T$ da matriz A dada por:

$$A = \begin{bmatrix} \sum_{i=1}^n \Delta X_{NR}^2 & \sum_{i=1}^n \Delta X_{NR} \Delta X_G \\ \sum_{i=1}^n \Delta X_{NR} \Delta X_G & \sum_{i=1}^n \Delta X_G^2 \end{bmatrix} \quad (8)$$

O método consegue excelentes resultados experimentais com uma baixa carga computacional. Assim, no Exemplo 1, converge em 3 iterações com o menor de todos os resíduos.

3 Fluxo de Potência em Redes Elétricas

O problema do Fluxo de Potência em Redes Elétricas consiste essencialmente na determinação do estado da rede (tensões em cada barra). O problema já foi amplamente tratado

na literatura [6, 10]. Do ponto de vista computacional, o problema consiste em calcular as tensões E conforme à equação:

$$YE = I(E), \quad \text{onde } Y \in \mathbb{C}^{N \times N}, \text{ e } E, I \in \mathbb{C}^N \quad (9)$$

I representa a corrente de cada barra, enquanto Y é conhecida como Matriz Admitancia. Esta matriz, além de ser simétrica, é altamente esparsa [7].

A tensão E_m da barra m , por ser complexa, pode ser expressa em coordenadas cartesianas e polares como:

$$E_m = Vr_m + jVi_m = V_m \exp(j\Theta_m) \quad (10)$$

Para a solução do Problema do Fluxo de Potência existem diversos métodos [6, 10]. Dentre todos eles o mais utilizado é o Desacoplado Rápido [6], uma sofisticação do Newton-Raphson. Uma das características mais interessantes deste método é a de considerar de forma desacoplada os valores de V e Θ . O outro método a ser combinado será o Jacobi [5], uma versão do Método da Matriz Y apresentada em [10]. O *Algoritmo Combinado* resultante, à semelhança da equação 2, está dado por:

$$E(k+1) = E(k) + \alpha(k) \Delta E_{DR}(k) + \beta(k) \Delta E_J(k) \quad (11)$$

ou de forma equivalente:

$$E(k+1) = c(k) E(k) + \alpha(k) E_{DR}(k) + \beta(k) E_J(k) \quad (12)$$

onde: $c(k) = 1 - \alpha(k) - \beta(k)$.

Fazemos notar que a tensão $E_{DR}(k)$ calculada pelo Método Desacoplado Rápido está naturalmente em coordenadas polares, enquanto $E_J(k)$ calculado pelo Método de Jacobi, em coordenadas cartesianas. Por simplicidade, supomos nesta seção que os pesos são reais e constantes em cada iteração ($\alpha(k) = \alpha$ e $\beta(k) = \beta$). Assim, a equação complexa 12 pode ser re-escrita em suas componentes cartesianas:

$$\begin{cases} Vr(k+1) = cVr(k) + \alpha Vr_{DR}(k) + \beta Vr_J(k) \\ Vi(k+1) = cVi(k) + \alpha Vi_{DR}(k) + \beta Vi_J(k) \end{cases} \quad (13)$$

A equação 13 pode ser ligeiramente modificada de forma a obter uma versão similar em coordenadas polares, dado que as barras PV são mais facilmente tratadas quando a tensão V_m aparece explicitamente.

$$\begin{cases} V(k+1) = cV(k) + \alpha V_{DR}(k) + \beta V_J(k) \\ \Theta(k+1) = c\Theta(k) + \alpha \Theta_{DR}(k) + \beta \Theta_J(k) \end{cases} \quad (14)$$

É importante salientar que as equações 13 e 14 não são em geral idênticas, mas estão muito próximas na medida que: $\Theta(k) \approx \Theta_{DR}(k) \approx \Theta_J(k)$

Os mapas obtidos utilizando coordenadas cartesianas e polares são apresentados em [1], onde pode-se notar que o número de iterações I é praticamente o mesmo para os dois casos encontrando-se algumas diferenças só no resíduo final. As mesmas características dos *mapas*, discutidas na seção 2, podem ser novamente verificadas para o problema do Fluxo de Potência.

4 Algoritmos Combinados de Pesos Variáveis em Problemas de Fluxo de Potência

Esta seção apresenta algumas metodologias de pesos variáveis usando minimização do resíduo. A equação 13 para coordenadas cartesianas, poderia ser re-escrita como:

$$\begin{cases} \dot{V}r(k+1) = c_1(k)Vr(k) + \alpha_1(k)Vr_{DR}(k) + \beta_1(k)Vr_J(k) \\ Vi(k+1) = c_2(k)Vi(k) + \alpha_2(k)Vi_{DR}(k) + \beta_2(k)Vi_J(k) \end{cases} \quad (15)$$

onde os pesos são agora função da iteração k e $c_i(k) = 1 - \alpha_i(k) - \beta_i(k)$, como já fora definido na equação 12.

A simplicidade do método da pseudoinversa quando usado com números reais, sugere a idéia de aplicar-lo separadamente para cada componente cartesiana da equação 15, mesmo que estas componentes não sejam independentes. O algoritmo assim resultante será chamado de XY .

Uma idéia que a priori parece melhor, é a utilização de coordenadas polares, pois já foi mencionado que para muitos problemas práticos as componentes polares são quase independentes. Isto pode ser feito modificando ligeiramente a equação 14, e utilizando separadamente a Pseudoinversa para cada componente. Este algoritmo será chamado de $V\Theta$:

$$\begin{cases} V(k+1) = c_1(k)V(k) + \alpha_1(k)V_{DR}(k) + \beta_1(k)V_J(k) \\ \Theta(k+1) = c_2(k)\Theta(k) + \alpha_2(k)\Theta_{DR}(k) + \beta_2(k)\Theta_J(k) \end{cases} \quad (16)$$

Para minimizar o Erro dado pela equação 7, para o caso complexo, uma vez mais temos que utilizar a Pseudoinversa A^+ , que para o caso complexo está dado por $A^+ := (A^H A)^{-1} A^H$. Esta proposta será denotada como algoritmo A^+ .

Uma última proposta analisada neste trabalho consiste em tentar minimizar de forma aceitável o resíduo E . Este algoritmo será denotado como S_∞ .

Os resultados experimentais destas 4 propostas de *Algoritmos Combinados de Pesos Variáveis* se resumem na Figura 2, onde são comparados aos outros algoritmos de *Pesos Constantes*, como o Jacobi (J), o Desacoplado Rápido (DR), o melhor *Team Algorithm* denotado como (TEAM) e o resultado utilizando o Desacoplado Rápido Sobre-Relaxado (DR-SR). Os resultados da Figura 2 foram obtidos para os seguintes *Problemas Tipo*:

- (1) IEEE 5-BUSBAR SYSTEM, sistema com 4 barras PV.
- (2) Sistema de 5 barras modificado, com 1 barra PV.
- (3) IEEE 6-BUSBAR SYSTEM, sistema sem barras PV.
- (4) Sistema modificado de 6 barras.
- (5) IEEE 14-BUSBAR SYSTEM, com uma barra PV.
- (6) Sistema de 14 barras, de alta carga.
- (7) Sistema crítico de 14 barras, altamente sobrecargado.
- (8) Sistema de 30 barras, exemplo de Monticelli (PWRD-86).
- (9) IEEE 57-BUSBAR SYSEM, com 6 barras PV.

Problema Tipo	Team Algorithms							
	Pesos Constantes				Pesos Variáveis			
	J	DR	DR-SR	TEAM	XY	V Θ	A ⁺	S _∞
1	3(45)	3(96)	2(51)	2(06)	2(30)	2(31)	*	2(23)
2	9(70)	3(27)	3(78)	2(88)	2(88)	2(88)	3(01)	2(58)
3	9(87)	8(89)	5(96)	5(96)	4(38)	6(80)	5(66)	7(58)
4	*	20(80)	16(79)	11(99)	*	20(97)	*	18(99)
5	*	4(11)	4(56)	3(82)	4(79)	4(52)	3(60)	3(43)
6	9(48)	24(83)	12(47)	8(56)	6(69)	5(61)	7(30)	7(61)
7	28(94)	35(95)	39(96)	24(89)	*	*	11(85)	15(62)
8	*	3(63)	4(64)	3(63)	4(18)	3(90)	4(08)	4(05)
9	*	4(30)	4(44)	4(30)	4(09)	4(08)	4(26)	4(83)

Figura 2: Comparação do número de iterações I requerido por cada algoritmo estudado para diferentes problemas tipo de Fluxo de Potência. Não-convergência em 100 iterações se denota com *. O resíduo na solução, multiplicado por 10^5 , é indicado entre parêntesis.

Na Figura 2 pode-se observar que nos primeiros 8 problemas tipo alguma versão do *Algoritmo Combinado* consegue superar qualquer um dos algoritmos sendo combinados, sendo o problema tipo No. (7) o mais significativo, pois o Jacobi converge em 28 iterações e o Desacoplado Rápido em 35, enquanto o *Team Algorithm* usando a Pseudoinversa Complexa consegue convergir em 11 iterações. O único caso onde nenhum ganho foi conseguido foi no problema tipo No. (8).

5 Conclusões

Os *Team Algorithms* apresentados na seção 2 resultaram promissores nos problemas que podemos chamar de ‘difíceis’, onde um ou ambos algoritmos da combinação falham em certas regiões por onde o algoritmo deve passar para chegar até a solução. Assim, o Exemplo 1 mostrou que é possível não só um menor número de iterações até a convergência, mais também ampliar o domínio de convergência.

Os resultados para o Fluxo de Potência resumidos na Figura 2 demonstram a potencialidade dos *Team Algorithms*, permitindo obter uma melhor convergência em 8 dos 9 casos testados. No entanto, o domínio de convergência não foi ampliado. Isto porque a condição inicial para o Fluxo de Potência está relativamente perto da solução numa região bem comportada e o algoritmo Desacoplado Rápido converge sem problemas.

Para concluir, enfatizamos o crescente interesse nos *Team Algorithms*; como exemplo, é discutida em [8, 9] a possibilidade de combinar algoritmos tradicionais con novas técnicas de inteligencia artificial [4] como os *algoritmos genéticos*.

Referências

- [1] Barán B. *Algoritmos Combinados. Uma aplicação para Fluxo de Potência Elétrica*. Relatório Técnico. Laboratório de Computação Paralela. COPPE. Universidade Federal de Rio de Janeiro, Brasil, 1991.
- [2] Bertsekas, D. P. e Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall Inc., 1989.
- [3] Borges C.L. *Investigação do Desempenho de Métodos de Solução do Fluxo de Potência para Processamento Paralelo e Vetorial*, COPPE/UFRJ. Tese M.Sc. Engenharia Elétrica. Rio de Janeiro, 1991.
- [4] Davis L. *Genetic Algorithms and Simulated Annealing*, Research Notes in Artificial Intelligence. Morgan Kaufmann Publishers Inc., Los Altos, California, 1987.
- [5] Dusonchet Y. P., Taludkar S. N. e Sinnot H. E. *Load Flows using a combination of Point Jacobi and Newton's Methods*, IEEE Transactions on Power Apparatus and Systems, Vol. PAS-90, pág. 941-949. Julho, 1971.
- [6] Monticelli A. *Fluxo de Carga em Redes de Energia Elétrica*, CEPEL. Editora Edgar Blücher Ltda. 1983.
- [7] Morozowski M. *Matrizes Esparsas em Redes de Potência: Técnicas de Operação*, Livros Técnicos e Científicos Editora S.A. Eletrobrás. Rio de Janeiro, 1981.
- [8] Ramesh V.C., Quadrel R., de Souza P. e Taludkar N. *Asynchronous Teams*, 1991 Summer National Meeting, American Institute of Chemical Engineers, Pittsburgh - U.S.A., 1991.
- [9] Souza P.S. e Taludkar S.N. *Genetic Algorithms in Asynchronous Teams*, Proceedings of the fourth international conference on Genetic Algorithms. San Diego, California, 1991.
- [10] Stott B. *Review of Load Flow Calculation Methods*, Proceeding of IEEE, Vol. 62, No. 7, Julho 1974.
- [11] Strang G. *Linear Algebra and its Applications*, Segunda edição. Academic Press Inc. Florida, 1980.
- [12] Taludkar S.N., Pyo S.S. e Mehrotra R. *Designing Algorithms and Assignments for Distributed Processing*, Electric Power Reserch Institute. Final Report. Preparado em a Universidade de Carnegie-Mellon, Pittsburg, Pennsylvania. Novembro, 1983.