Determinação do Ranking de Contingências em Sistemas de Energia Elétrica Utilizando MPI (*Message Passing Interface*)

OMAR ANDRES CARMONA CORTES*
OSVALDO RONALD SAAVEDRA MENDEZ+

* USP - Universidade de São Paulo
ICMC - Inst. de Ciências Matemáticas e de Computação
Secretaria de Pós-Graduação
Caixa Postal 668
13560-970 São Carlos - SP
ocortes@icmc.sc.usp.br

+ UFMA – Universidade Federal do Maranhão
DEE – Departamento de Engenharia de Eletricidade
Av. dos Portugueses s/n Campus Universitário
65080-040 - São Luis – MA
osvaldo@bacuri.dee.ufma.br

Resumo: Neste trabalho apresenta-se uma proposta para a determinação do ranking de contingências para sistemas de energia elétrica utilizando processamento paralelo. Para explorar adequadamente o ambiente computacional, são utilizadas as metodologias Gauss-Seidel e Jacobi sob a abordagem de programação SPMD (*Simple Program Multiple Data* - Programa Simples Múltiplos Dados). A idéia não é paralelizar esses algoritmos e sim resolver em paralelo vários cenários de contingências. Os algoritmos foram testados em um computador paralelo IBM-SP2 com três e dois processadores, utilizando um sistema de transmissão brasileiro. São discutidos aspectos relacionados com o desempenho e são reportados os resultados práticos obtidos com o algoritmo.

Palavras-Chave: análise de segurança, programação paralela, ranking de contingências, análise de desempenho, *speedup*, eficiência.

1. Introdução

Existem muitos problemas que podem ser modelados com sistemas lineares ou através de modelos linearizados. Desses problemas, muitos apresentam um alto grau de esparsidade, principalmente os problemas relacionados a sistemas de energia elétrica, além de que, esses sistemas apresentam elevadas dimensões.

Os métodos iterativos foram intensamente utilizados em aplicações de potência durante as décadas dos 50 e 60 [1] e logo foram esquecidos com o surgimento de outros métodos tais como fatoração triangular usando esparsidade [2-4]. Apesar de serem confiáveis e robustos, as taxas demoradas de convergência foram o fator decisivo na sua substituição por estes últimos métodos. Porém, com a evolução dos computadores paralelos e com o alto desempenho dos processadores atuais, os métodos iterativos voltaram a ser considerados, em especial o método de Jacobi, devido ao paralelismo natural que esse método apresenta [5]. Apesar disso, os métodos numéricos utilizados não foram paralelizados, pois o método de Gauss-Seidel apresenta um nível de granulação[1] muito fino. O que foi plenamente paralelizado é a forma de calcular o ranking de contingência.

Por outro lado, o processo de seleção e ordenação de contingências é de fundamental importância no monitoramento de segurança em tempo real de um sistema de potência, para tomar decisões de controle,

[1] A granulação refere-se a forma de como uma tarefa é paralelizada. Granulação grossa indica tarefas grandes com pouca comunicação. Granulação fina indica muitas tarefas pequenas com muita comunicação. Pode-se dizer que a granulação média situa-se entre a fina e a grossa, pois é muito difícil quantificar esse parâmetro.

seja a critério do operador, seja através da determinação, uma função tipo fluxo de potência ótimo, de um ponto ótimo seguro [6] e [7]. O processo consiste na identificação de uma lista de casos que podem causar violação no sistema. Além do número de candidatos que podem causar esse problema ser grande, a exatidão no processo de seleção deve ser adequada e a velocidade de processamento deve ser alta.

Para tornar a seleção computacionalmente viável, normalmente a exatidão é sacrificada. Levando em consideração este aspecto, é necessário um esquema especial de ordenação para classificar em primeiro lugar as contingências de maior severidade, ou seja, colocar em primeiro lugar as que causam situações emergenciais mais delicadas. Os métodos de ordenação baseiam-se em um índice que quantifica a severidade da contingência. Estes índices são chamados de *índices de severidade* ou *índices de desempenho*.

Neste trabalho apresenta-se o cálculo desse índice utilizando os métodos de Gauss-Seidel e Jacobi. A vantagem deles é que se evita a refatoração da matriz para a simulação de cada caso de contingência. Do ponto de vista de processamento paralelo, não são paralelizados os métodos em si e sim os cenários de contingências, que em situações reais podem atingir um elevado número. Logo, tem-se um paralelismo de granulação grossa.

Este trabalho está organizado da seguinte forma: Primeiro são revisados algum aspecto do processamento paralelo, é apresentado uma pequena introdução sobre análise de desempenho e sobre as metodologias clássicas Jacobi e Gauss-Seidel. Os itens 4 e 5 fazem uma abordagem sobre ranking de contingência e como esse processo foi paralelizado. Finalmente, são realizados os testes utilizando a rede do sistema sul-sudeste. A plataforma computacional utilizada é um computador paralelo IBM-SP2 com três processadores.

2. Processamento paralelo

Processamento paralelo é uma tecnologia "moderna" que tenta suprir uma demanda por alta performance computacional. O paralelismo pode aparecer de diversas formas, tais como: *lookahead*, *pipelining*, vetorização, concorrência, computação distribuída, etc.[8]

Dentro da computação paralela existem duas abordagens de programação: O SPMD e MPMD.

O SPMD (*Simple Program Multiple Data* - Programa Simples Múltiplos Dados) utiliza o mesmo código de programa em diferentes dados. O MPMD (*Multiple Program Multiple Data* - Múltiplos Programas e Múltiplos Dados) utiliza diferentes programas lidando ao mesmo tempo com diferentes dados.

Após o desenvolvimento do programa paralelo é necessário calcular o desempenho do novo algoritmo dentro do sistema em questão. Os ganhos do programa paralelo em relação ao programa seqüencial são medidos através do *speedup* e da eficiência. Para comparação entre diferentes arquiteturas outras métricas de avaliação devem ser utilizadas como mostrado em [9].

O *speedup* é dado pela expressão:

$$Sp = \frac{T_1}{T_p}$$

T_1 é o tempo de execução do programa seqüencial e T_p é o tempo de execução do programa paralelo. Baseado ainda no *speedup* pode-se calcular a eficiência da implementação paralela através da expressão:

$$Ef = \frac{Sp}{Np}$$

Sp é o *speedup* alcançado e *Np* é o número de processadores utilizados para executar o programa paralelo.

Dada as fórmulas acima nota-se que o *speedup* ideal deve ser igual a quantidade de processadores utilizados no programa paralelo. A eficiência deve estar entre zero e um, pois indica um valor relativo. Se for alcançado um *speedup* ideal também é alcançada a eficiência ideal que é igual a um (indicando 100% de eficiência).

Em casos raros pode ocorrer a anomalia do *speedup* e conseqüentemente a anomalia da eficiência, onde os valores são superiores ao ideal em ambos os casos. Mais detalhes e casos reais de anomalia podem ser encontrados em [10].

3. Métodos iterativos

Existem muitos problemas que podem ser modelados com sistemas lineares ou através de modelos linearizados. Desses problemas, muitos apresentam um grau elevado de esparsidade. No caso de sistemas de energia elétrica (como se verá mais adiante), além de se observar um alto grau de esparsidade, as matrizes obtidas também apresentam elevadas dimensões e comumente são exigidas soluções repetidas, com exigência de tempo (em especial em aplicações em tempo real).

A classe dos métodos iterativos resolve o sistema convertendo um vetor $x^{(k)}$ em outro, $x^{(k+1)}$, que depende de $x^{(k)}$, A e b, e preservam a esparsidade da matriz A, pois seus elementos não são alterados. A idéia central

dos métodos Iterativos pode ser colocada na seguinte forma:

Seja o sistema linear Ax = b, onde:

A : matriz dos coeficientes, n x n;

x : vetor de variáveis, n x 1;

b : vetor dos termos constantes, n x 1.

É, então, proposto o seguinte esquema iterativo:

Inicia-se o passo em $x^{(0)}$, onde $x^{(0)}$ é o vetor de aproximação inicial. De um modo geral, as próximas iterações (k + 1) são dependentes da iteração no passo k. O item 2.2 detalha esse processo com um pouco mais de detalhes.

3.1 Critério de parada

O processo de iteração repete-se até que o vetor $x^{(k+1)}$ esteja próximo do vetor $x^{(k)}$, dado um determinado erro ou precisão (ε) de modo que $M^{(k)} < \varepsilon$, onde:

$$M_R^{(k)} = \frac{M^{(k)}}{máx_{1,2,\ldots,n} \left| x_i^{(k)} \right|}$$

onde $M^{(k)}$ corresponde a maior diferença de $\left| x_n^{(k+1)} - x_n^{(k)} \right|$, e o denominador é o maior valor em módulo do vetor $x^{(k+1)}$.

A seguir serão rapidamente revisados os métodos iterativos, Gauss-Seidel e Jacobi.

3.2 Método iterativo de Jacobi

A forma como o método de Jacobi soluciona o sistema linear Ax=b é descrito a seguir.

Consideremos o seguinte sistema:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{13}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{23}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + \cdots + a_{33}x_n = b_3 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{n3}x_n = b_n \end{cases}$$

Isolando o termo x_i, onde, i = 1,2,...,n, em cada equação tem-se:

$$\begin{cases} x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n) \\ x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n) \\ \vdots \\ x_n = \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1}) \end{cases}$$

O método de Jacobi consiste em dado uma aproximação inicial $x^{(0)}$, encontrar $x^{(1)},\ldots,x^{(k)}$, através de uma relação recursiva, ou seja, usar o valor de $x^{(k)}$ para encontrar a próxima aproximação $x^{(k+1)}$. As equações resultantes são:

$$\begin{cases} x_1^{(k+1)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \ldots - a_{1n}x_n^{(k)}) \\ x_2^{(k+1)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \ldots - a_{2n}x_n^{(k)}) \\ \vdots \\ x_n^{(k+1)} = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \ldots - a_{n(n-1)}x_{n-1}^{(k)}) \end{cases}$$

Onde a aproximação inicial $x^{(0)}$, que será usada na primeira iteração corresponde a um vetor de zeros.

3.3 Método de Gauss-Seidel

O método de Gauss-Seidel é semelhante ao método de Jacobi. A diferença está em que o método para obtenção das iterações é baseado na iteração imediatamente anterior.

Portanto no método de Gauss-Seidel, no momento de calcularmos $x_j^{(k+1)}$, com i=2,3,...,n, usamos todos os valores já encontrados na iteração $x_j^{(k+1)}$. O método pode ser visto com detalhes em [15] e [16].

4. Análise de segurança em sistemas de energia elétrica

A análise de segurança tem duas funções. A primeira é identificar se existe alguma violação de limites no estado operativo atual. Este processo, na sua forma mais simples, envolve o monitoramento de fluxos de potência, tensões, etc. e sua comparação com limites operacionais pré-especificados.

A segunda é identificar as violações que ocorrerão em caso de perda de um ou mais componentes do sistema, tais como: linhas, transformadores, geradores, etc. Nesta função, também conhecida como **Análise de Contingência**, poderá ser simulada a perda de componentes através de modelos dinâmicos (estabilidade transitória) ou estáticos (fluxo de carga).

O processo de contingência é realizado através de uma lista de prováveis acontecimentos que podem levar a situações emergenciais muito graves. Dependendo dos critérios de operação e da severidade do problema, haverá uma resposta a essa lista. Devido a essa resposta pode-se tomar três tipos de atitudes, a seguir descritas:

- Alterar o estado de pré-contingência para aliviar ou evitar a emergência resultante.

- Realizar uma estratégia que simplesmente alivie a emergência.
- Nada fazer se a emergência ocorrida for mínima ou improvável de acontecer.

4.1 Índices de severidade

O índice de severidade (ou de desempenho) é um número que quantifica a severidade da contingência. Existem vários tipos de índices; o mais popular é o índice baseado na sobrecarga, ou seja, que mede qual será a sobrecarga total de um sistema caso um componente seja retirado. O índice baseado na sobrecarga, para contingências de potência ativa é dado por:

$$PI = \sum_{j=1}^{m} \left(\frac{\omega_j}{2n} \right) \left(\frac{T_j}{T_j^{max}} \right)^{2n}$$

onde T_j é o fluxo de potência ativa no ramo j; T_j^{max} é a capacidade de transmissão do ramo, ω_j é um fator de ponderação m é o número de ramo j, e n é um inteiro. No caso estudado os parâmetros ω_j e n foram considerados unitários. Note que m não inclui o ramo retirado para simular a contingência.

5. Paralelização do processo de determinação do ranking de contingências

Para realizar a paralelização dos cálculos de contingência foi utilizado o MPI (*Message Passing Interface*) que é uma extensão paralela de linguagens seriais, ou seja, é uma biblioteca que pode ser encontrada em Fortran, C e C++ [11]. Neste projeto foi utilizada a linguagem C.

A abordagem de programação utilizada é a SPMD onde todos os processadores têm o mesmo código de programa, mas trabalham em dados diferentes. O paradigma utilizado é o mestre-escravo com passagem de mensagem onde um processador é designado como mestre e controla os dados que devem ser passados para os demais processadores (escravos). Em [8], [12] e [13] existem mais informações sobre paradigmas e abordagens de programação paralela.

No ambiente sul-sudeste são analisadas 910 quedas de linhas. Esse número é dividido pela quantidade de processadores e cada um deles se responsabiliza por uma determinada quantidade de quedas de linha. Por exemplo:

Para 3 processadores tem-se:

$$nl = \frac{910}{3} = 303$$

O resto desse cálculo é adicionado ao mestre, dessa forma, cada escravo soluciona 303 linhas e o mestre soluciona 304.

As mensagens passadas são apenas duas: a quantidade de linhas que cada processador deve solucionar e um *flag* para indicar ao mestre que os escravos terminaram o processamento. Isso dá um alto grau de granulação ao programa.

6. Resultados dos testes

A seguir são apresentados resultados com um sistema real de 1658 barras e 2365 ramos correspondente à rede brasileira sul-sudeste de transmissão de energia elétrica.

As simulações foram realizadas num computador multiprocessado IBM-SP2 cujas características são: 3 processadores PowerPC com 66.7 Mhz, sendo 1 do tipo *wide* e 2 de tipo *thin*; todos com 256 Kbytes de cache e 9 Gigabytes de disco.

Nas Figuras 1 e 2 o eixo-x representa a tolerância de convergência ou precisão e o eixo-y representa o tempo de execução em segundos.

A Figura 3 segue a seguinte legenda:

I. *Speedup* alcançado em Gauss-Seidel em 2 processadores
II. *Speedup* alcançado em Jacobi em 2 processadores
III. *Speedup* alcançado em Gauss-Seidel em 3 processadores
IV. *Speedup* alcançado em *Jacobi* em 3 processadores

A Figura 4 segue a mesma legenda da Figura 3, exceto pelo fato de que no lugar de *Speedup* é apresentada a eficiência alcançada.

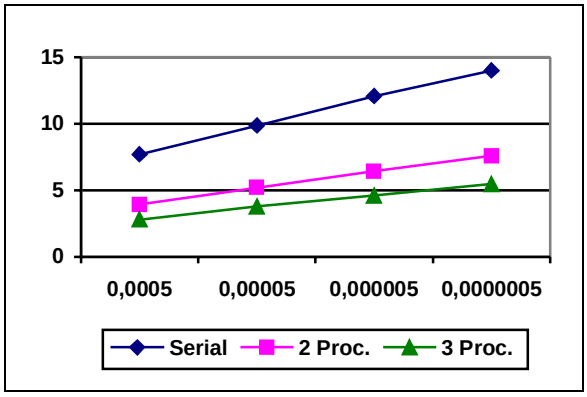

Figura 1 - Gráfico de tempo de execução utilizando o método de Gauss-Seidel

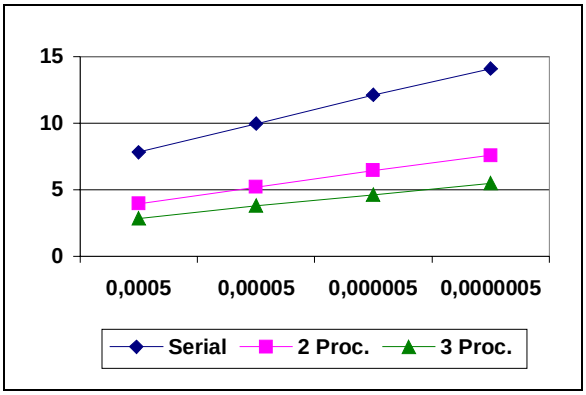

Figura 2 - Gráfico de tempo de execução utilizando o método de Jacobi

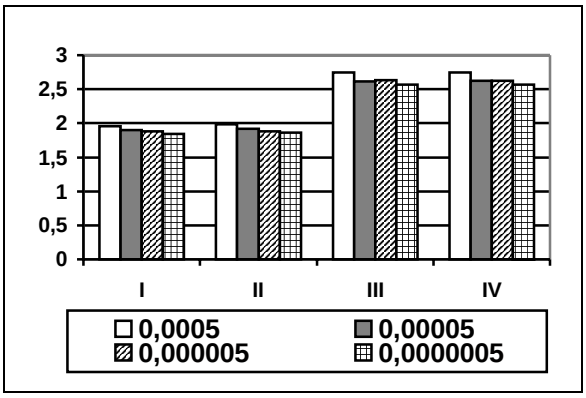

Figura 3 - Gráfico de Speedup alcançados em 2 e 3 processadores

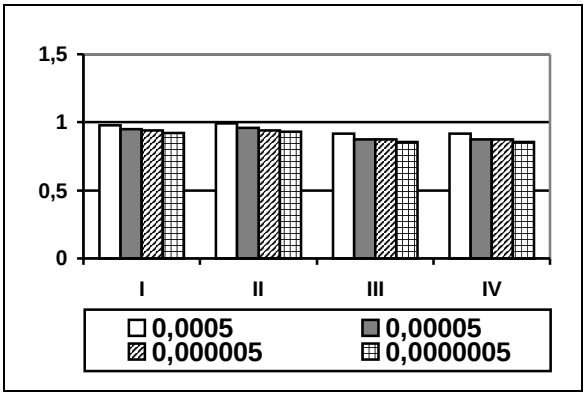

Figura 4 - Eficiência alcançada por 2 e 3 processadores

Comentários:

Os resultados obtidos mostram um ganho razoável no desempenho do algoritmo quando este é executado em um computador paralelo conforme mostram as Figuras 1 e 2. Devido a grande esparsidade da matriz e da forma como ela é armazenada (baseada no algoritmo de Zollenkopf[2] [4]), a diferença em tempo de execução entre os dois métodos é muito pequena por isso os gráficos das Figuras 1 e 2 são semelhantes.

Observa-se que o *speedup* alcançado é quase o ideal quando a precisão dos cálculos não é grande. Nota-se também, que o *speedup* diminui à medida que a precisão dos cálculos aumenta, mas a queda não é acentuada se comparada ao grande aumento das precisões.

Devido ao fato do *speedup* ser quase o ideal a eficiência alcançada também é quase a ideal, sendo que a eficiência para 2 processadores é um pouco maior em conseqüência a Lei de Ahmdal[3]. Mais detalhes sobre a Lei de Ahmdal podem ser vistos em [8], [10], [12] e [13].

Nota-se também nas Figuras 1 e 2, que o único valor que é alterado na execução do programa é a precisão dos cálculos. Como resultado do aumento na tolerância tem-se um aumento no tempo de processamento. A diferença no tempo de execução é da ordem de um segundo a medida que aumenta a precisão dos cálculos.

7. Conclusões

Neste trabalho foi apresentada uma proposta para a determinação do ranking de contingências para sistemas de energia elétrica, utilizando processamento paralelo através do MPI. Para explorar adequadamente o ambiente computacional, foram utilizadas as metodologias Gauss-Seidel e Jacobi paralelizando o cálculo do ranking de contingência o que permitem uma excelente taxa de paralelização. O algoritmo foi testado em um computador Paralelo IBM-SP2, utilizando um sistema de transmissão Brasileiro, obtendo-se bom ganho como se pode observar dos resultados obtidos e reportados aqui.

8. Referências

[1] Ward, J. e Hale, W., "**Digital Computer Solution of Power Flow Problems**", AIEE Transactions Power App. Syst., Vol PAS-75, junho 1956, pp. 398- 404.

[2] Tinney, W. F. and Walker, J. W., "**Direct Solutions of Sparse Network Equations by Optimally**

[2] Método baseado em vetores no qual uma matriz esparsa é armazenada sem os elementos nulos.

[3] A Lei de Ahmdal - à medida que processadores são adicionados ao processamento, o *speedup* e a eficiência tendem a cair. A partir de um número n de processadores essa queda é extremamente brusca.

Ordered Triangular Factorization'', Proceedings of IEEE, Vol 55, novembro 1967, pp. 1801- 1809.

[3] Tinney, W. F. and Hart, C. E. , "**Power flow Solution by Newton's Method**", IEEE Transactions Power App. Syst., Vol PAS-86, novembro 1967, pp. 1449 - 1460.

[4] Zollenkopf. K., ``**Bi-factorisation Basic Computacional Algorithm and Programming Techniques**", In: Reid. J. K. (Editor), "**Large Sparse Sets of linear Equations**'', Academic Press, pp 75-96, 1971".

[5] La Scala, M. et. Al., "**Gauss-Jacob-Block-Newton Method for Parallel Transient Stabulity Analysis**", IEEE Trans. on Power Systems, Vol PWRS-5, No 4, pp.1168-1177, nov. 1990.

[6] M. Rodrigues, O. R. Saavedra, A. Monticelli, "**Asynchonous Programming Model for the Concurrent Solution of Security Constrained Optimal Power Flow Problem**", IEEE Transaction on Power Systems, Vol 9 No 4 novembro 1994.

[7] Saavedra, O.R., "**Solving the Security Constrained Optimal Power Flow on a Distributed Processing Environment**", Proceeding of IEE -Part C, nov. 1996.

[8] Hwang, Kai., "**Advanced Computer Architecture: parallelism, scalability, programability**",McGraw-Hill, Illinois, 1993.

[9] Hockney, R. W., "**The Science of Computer Benchmarking**", Philadelphia: society for Industrial and Applied Mathematics, 1996.

[10] Carmona Cortes, O. A., "**Desenvolvimento e Avaliação de Algoritmos Numéricos Paralelos**", Dissertação de Mestrado, ICMC/USP, São Carlos - SP, 1999.

[11] MacDonald, N., Minty, E., Harding, T., Browa, S., "**Writing Message-Passing Parallel Programs with MPI – Course Notes**". The University of Edinburgh, 1994.

[12] Almasi, G. S., Gottilieb, A., "**High Parallel Computing**". 2ª ed, The Benjamin Cummings Publishing Company, Inc., 1994.

[13] Quinn, M., J., "**Designing Efficient Algorithms for Parallel Computers**", McGraw Hill, New York, 1997.

[15] Ruggiero, M. A. e Lopes, V. L. da R. "**Cálculo Numérico: Aspectos Teóricos e Computacionais**", São Paulo: McGraw-Hill, 1996.

[16] Claudio, D. C. e Marins, J.M. "**Cálculo Numérico Computacional: Teoria e Prática**",2ª ed, São Paulo: Atlas, 1994..

Agradecimentos

Ao CNPq (Conselho Nacional de Pesquisa);

A FAPEMA (Fundação de Amparo a Pesquisa do Maranhão);

A CESUP/UFRGS (Centro de Supercomputação da Universidade Federal do Rio Grande do Sul). Ao CISC/USP (Centro de Informática de São Carlos da Universidade de São Paulo).

E ao LASD-PC/USP - ICMC (Laboratorio de Sistemas Distribuídos e Programação Concorrente)

Sem as quais este trabalho não seria possível.

Biografias

OMAR A CARMONA CORTES (MSC99; BSC97) *é* aluno especial de doutorado do programa de pós-graduação do ICMC/USP (Instituto de Ciências Matemáticas e de Computação/ Universidade de São Paulo) na área de Sistemas Distribuídos e Programação Concorrente.

OSVALDO R. SAAVEDRA (DSC, 93; MSC, 88; EE, 81) é professor do Departamento de Engenharia Elétrica da UFMA.