DSEE

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Sistemas de Energia Elétrica

UNICAMP

Processamento Distribuído Aplicado à Análise de Segurança Estática de Sistemas de Energia Elétrica

Autor: Antônio César Baleeiro Alves

Orientador: Professor Dr. Alcir José Monticelli

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da UNICAMP, como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

Campinas, Agosto de 1997

Este exemplar corresponde a redação final da tese defendida por Antonio Cesar Baleeiro Alves e aprovada pela Comissão Julgada em 15 / 08 / 97

Orientador

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

AL87p

Alves, Antônio César Baleeiro Alves
Processamento distribuído aplicado à análise de segurança estática de sistemas de energia elétrica / Antônio César Baleeiro Alves.--Campinas, SP: [s.n.], 1997.

Orientador: Alcir José Monticelli.
Tese (doutorado) - Universidade Estadual de Campinas Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de energia elétrica. 2. Processamento distribuído. 3. Processamento paralelo (Computadores). 4. Otimização matemática. 5. Métodos do gradiente conjugado. I. Monticelli, Alcir José. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Tese apresentada e aprovada em 15 de agosto de 1997, na Sala de Reuniões da CPG/FEEC/UNICAMP, perante a seguinte banca examinadora:

Alcir J. Monticelli, UNICAMP _____

André L. Morelato França, UNICAMP _____

Ariovaldo Verandio Garcia, UNICAMP _____

Armando M. L. da Silva, EFEI _____

Carlos Alberto Castro Jr., UNICAMP _____

Djalma M. Falcão, COPPE/UFRJ _____

Fujio Sato, UNICAMP _____

Dedico este trabalho:

Aos meus queridos pais,

Edy Alves Baleeiro e
Antônio Rocha Baleeiro.

AGRADECIMENTOS

A Deus que me concedeu vida e saúde

à minha esposa, Maria Abadia Freire Baleeiro, grande companheira de todos os momentos; aos meus filhos, Flávio e André, motivações da minha existência

aos tios Aloísio e Armando, e aos avós paternos, pelo extraordinário apoio nos primeiros dias da minha formação

ao Dr. Alcir José Monticelli pela excelente orientação, incentivo, e pelos tantos desafios propostos

ao amigo Fujio Sato, pessoa do mais alto valor com quem muito aprendi, pela inestimável colaboração

ao Professor Ariovaldo pelas valiosas ajudas

ao Professor André pelo incentivo e pelas primeiras lições de processamento paralelo

à secretária do DSEE, Senhorita Edna

ao jovem amigo Eduardo Nobuhiro Asada pela proveitosa colaboração e amizade sincera

ao amigo Sérgio Ricardo pelo incentivo e apoio

aos amigos e colegas de trabalhos conjuntos, Ramon e Ruben Romero

aos amigos de Juiz de Fora, Eduardo Nicola, Francisco Eugênio e Marcelo pela colaboração

ao Leonardo Paucar e ao Sérgio Azevedo pela amizade sincera

aos amigos e colegas do Laboratório (LSEE), em especial ao Mantovani, José Vicente, Aleardo, Miguel, Haffner, Fernando, Regina e Luciana pelo companheirismo e atenção

a Idalina, nossa competente secretária em Goiânia

aos Professores da Escola de Engenharia Elétrica da UFG, em especial ao Emilson, Bernardo, Enes, Adalberto José Batista, Paulo César e Euler pelo incentivo e ajuda

ao pessoal da Pró-Reitoria de Pesquisa e Pós-Graduação da UFG, em especial ao Prof. Valter Casseti e às servidoras Santina e Marilene

aos ex-alunos de Goiânia reencontrados em Campinas

aos dedicados ex-professores de tantos cursos

às instituições CAPES, FAPESP e CENAPAD-SP pelo apoio financeiro ou pela disponibilização de equipamentos

O olho do poeta, rolando num frenesi,
Vai da terra para o céu, do céu para a terra;
E enquanto a imaginação vai dando corpo
A formas de coisas desconhecidas, a pena do poeta
Transforma-as em figuras e dá ao nada etéreo
Um nome e um lugar para morar.

Shakespeare

Resumo

Este trabalho focaliza o problema de análise de contingências e as técnicas de cálculo do fluxo de potência ótimo com restrições de segurança.

A abordagem do fluxo de potência ótimo está essencialmente voltada para o alívio de violações de limites de fluxos de potência ativa em ramos (sobrecargas) através de ações sobre os controles ativos. O enfoque fundamenta-se na linearização do modelo do sistema elétrico, utiliza a técnica de programação linear e algoritmos de análise de contingências. As restrições de segurança são geradas a partir de uma classificação prévia das contingências para identificar os casos críticos; são incorporadas capacidades corretivas pós-contingências e a construção das restrições baseia-se nas metodologias de Stott e Benders.

Quanto à análise de contingências, foram investigadas duas classes de métodos, os diretos e os iterativos. Melhoramentos foram introduzidos nos métodos iterativos do gradiente conjugado pré-condicionado (como a ordenação *minimum spanning tree* e um pré-condicionador adequado ao problema de análise de contingências) para torná-los competitivos com os métodos diretos; contingências simples e múltiplas são analisadas.

Por meio do sistema *PVM*, as implementações desenvolvidas nesta tese foram transportadas para um ambiente de processamento distribuído, ou seja, estações de trabalho interligadas em rede local, formando uma máquina paralela virtual. Foi também pesquisada a elaboração de programas tolerantes a falha.

Foram realizados vários testes e simulações com implementações seqüenciais e distribuídas para validar os algoritmos elaborados, inclusive com dados do Sistema Interligado Brasileiro.

Abstract

This work is focused on the contingency analysis problem and the techniques involved with the security constrained optimal power flow calculation.

The optimal power flow approach is essentially directed to alleviate the violations of the active power flows in branches (overloads) through actions over the active controls. This approach is based in the linearization of the electric system model, using linear programming techniques and contingency analysis algorithms. The security constraints are generated from a previous contingencies classification in order to identify the critical situations; post-contingency corrective capabilities are incorporated and the Stott and Benders methodologies are used to build up such restrictions.

Two contingency analysis classes of methods were investigated: the iterative and the direct methods. Improvements were introduced into pre-conditioned conjugate gradient iterative method (the minimum spanning tree ordering and a pre-conditioner adequated to the contingency analysis for example) in order to make them as well efficient as the direct methods are; simple and multiple contingency are analyzed.

The implementations developed in this thesis were ported to a distributed processing environment through the PVM system, creating a virtual parallel machine with several workstations connected through a local network. The fault-tolerant approach is also investigated.

Several simulations and tests were performed, using sequential and distributed programs, to validate the proposed algorithms, including the Brazilian Interconnected Power System data.

Conteúdo

NOTAÇÕES, CONVENÇÕES E ABREVIATURAS

Notações e Convenções

- Termos estrangeiros são grafados em itálico. Por exemplo, *software*, *shunt*, *ranking*, etc.
- Termos figurativos, usuais no dia a dia, empregados como recurso explicativo no texto, são grafados entre aspas. Por exemplo, "casada", "janela", "infactibilidade", "pesada", etc.
- Vetores são simbolizados por letras maiúsculas ou minúsculas em negrito. Por exemplo,

  $\mathbf{e}_{ij}$, $\mathbf{v}_{\alpha_{km}}$, $\mathbf{A}^{t}_{km}$, etc.
- Matrizes são simbolizadas sempre por letras maiúsculas entre colchetes ([ ]). Por exemplo,

  $[B']$, $[M]$, $[A]$, etc.
- A transposta de uma matriz (ou transposto de um vetor) é simbolizada com o superíndice t.
- Nas equações onde está escrito

  s.a. significa sujeito a;

  Min significa minimizar;

  Max significa maximizar;

  Mín significa mínimo;

  Máx significa máximo.

Abreviaturas

- BR - Brasil (por exemplo, BR-725 é um banco de dados do Sistema Interligado Brasileiro, com 725 barras).
- CAG - Controle Automático de Geração
- *CDF - Commom Data Format*
- *e.g. - exempli gratia* (por exemplo)
- *EMS - Energy Management System*
- *Ethernet* - Marca registrada da *XEROX Co.*
- *FDDI - Fiber Distributed Data Interface*

- FPO - Fluxo de Potência Ótimo
- FPORS - Fluxo de Potência Ótimo com Restrições de Segurança
- GFLOPS - 10^9 operações de ponto flutuante por segundo
- *GMRES - Generalized Minimum Residual*
- IBM - *International Business Machine*
- *i.e.* - *id est* (isto é)
- *IEEE - Institute of Electrical and Electronics Engineers*
- *I/O - Input/Output*
- *LAN - Local Area Network*
- *MD - Minimum Degree*
- MDQ - Mínimo Desvio Quadrático
- MIPS - Milhões de instruções por segundo
- *ML - Minimum Lenght*
- *MPI - Message Passing Interface*
- *MST - Minimum Spanning Tree*
- *PCG - Pre-conditioned Conjugate Gradient*
- *PECO - Philadelfia Electric Co.*
- PL - Programação Linear
- *PVM - Parallel Virtual Machine*, *PVM3* - Versões 3.x do sistema *PVM*
- *RISC - Reduced Instruction Set Computer* (Computador com um conjunto reduzido de instruções)
- *SCADA - Supervisory Control And Data Acquisition*
- *SOR - Sucessive Overrelaxation*
- *SPMD - Single Program Multiple Data*, e *MPMD - Multiple Program Multiple Data*
- *Sun* - Marca registrada (e.g., estação de trabalho da *Sun*)
- UCP - Unidade Central de Processamento (*CPU*)
- *UNIX* - Sistema operacional que teve origem nos sistemas *UNIX-V* da AT&T e *4.2BSB*
- *VLSI - Very Large System Integration*

Capítulo 1

INTRODUÇÃO

Segurança Estática: Conceitos e Objetivos

O principal objetivo do planejamento e operação do sistema de energia elétrica consiste em atender a demanda de forma confiável, enquanto simultaneamente são respeitadas certas restrições de qualidade impostas a geração e a transmissão. Geralmente, os níveis especificados de confiabilidade e qualidade são assegurados em termos da adequação e da segurança do sistema. Adequação refere-se à capacidade do sistema de suprir as cargas, e segurança refere-se à habilidade para suportar o impacto de um distúrbio (contingência). O sistema é considerado seguro se nenhum limite de segurança for seriamente violado no evento de uma contingência. O processo de investigação para saber se o sistema estará seguro mediante um conjunto de contingências propostas é chamado de análise de segurança [27,30].

A segurança do sistema envolve projetos práticos para manter o sistema elétrico operando mesmo quando componentes falham. Por exemplo, uma unidade de geração pode ter que ser desligada por causa de uma falha do equipamento auxiliar. Pela manutenção de suficiente reserva girante, as unidades em operação poderão suprir o *déficit* sem precisar baixar a freqüência ou cortar carga. De maneira similar, uma linha de transmissão pode ser atingida por uma tempestade e ser desconectada pela ação automática da proteção. Como conseqüência desta falha, as linhas de transmissão vizinhas sofrerão aumento de carregamento e poderão ainda permanecer dentro dos limites.

O instante da ocorrência do evento que leva à falha de um equipamento é imprevisível. Por esta razão é que o sistema dever ser operado evitando-se sempre condições perigosas. Uma vez que os equipamentos do sistema de potência são projetados para operar dentro de determinados limites, algumas partes dos equipamentos são protegidas por dispositivos que os desconectam do sistema caso esses limites sejam violados. Se um evento qualquer ocorrer num sistema a ponto de deixá-lo operando com limites violados, tal evento poderá ser seguido por uma série de

ações adicionais de chaveamento de outros equipamentos. Se este processo de falhas em cascata prosseguir, o sistema como um todo (ou grandes partes dele) poderá entrar completamente em colapso. Isto é normalmente referido como ***blackout*** do sistema [59]. Três exemplos reais podem ser dados para ilustrar este tipo de ocorrência: (1) ***blackout*** de 1965 no Nordeste dos E.U.A.; (2) ***blackout*** ocorrido em abril de 1984 no Brasil; e (3) ***blackout*** na Costa Oeste dos E.U.A., em 1996. O primeiro tem grande importância histórica e é considerado um marco para a análise de segurança; está descrito a seguir. Um caso clássico de um grande ***blackout*** ocorreu nos Estados Unidos em 9 de outubro de 1965, que atingiu todo o estado de New York, os estados de Connecticut, Rhode Island, Massachusetts e Vermont, partes dos estados de New Hampshire, New Jersey e Pennsilvania, e a maior parte da província de Ontario no Canadá. Cerca de 30 milhões de pessoas ficaram privadas da energia elétrica ao mesmo tempo; uns por alguns minutos e outros por mais de 13 horas. Este grande desastre iniciou no final da tarde ($17 : 16 : 11$ $[h, min, s]$) de uma forma prosaica, ou seja, pela abertura de uma das cinco linhas de 230 $[kV]$, da subestação Sir Adam Beck da empresa Ontario Hydro, por um relé de retaguarda ajustado abaixo das condições de operação daquele ano. Assim, o fluxo de potência da linha desligada passou para outras quatro linhas remanescentes, provocando desligamentos em cascata, também pelos relés de retaguarda, em $2,7$ segundos, causando o que ficou conhecido como "o grande ***blackout*** de 65".

Em vista de fatos como o descrito acima, quase todos os sistemas de potência são operados de modo que qualquer simples falha inicial não sobrecarregue pesadamente outros equipamentos, especificamente para evitar falhas em cascata [59].

Um requisito chave de qualquer sociedade moderna é a operação econômica e segura do seu sistema de energia elétrica. Um objetivo tão importante naturalmente demanda o uso de avançadas análises de grandes sistemas, otimização, e tecnologia de controle. Esta tecnologia está sendo incorporada dentro das funções de economia-segurança dos Centros de Supervisão e Controle (***Energy Management System, EMS***, em inglês) das companhias de energia elétrica. O ***EMS*** é um grande e complexo sistema de ***hardware*** e ***software*** baseado no principal centro de controle da empresa [27].

A maioria dos grandes sistemas de potência possui equipamentos instalados para permitir ao pessoal da operação monitorar e operar o sistema de uma maneira confiável. As técnicas e os equipamentos usados nestes sistemas para este fim são comumente reunidas sob o título segurança do sistema.

A meta global do controle econômico-seguro é operar o sistema com o menor custo, com a garantia de evitar as condições de emergência ou de escapar delas. Esta meta implica na necessidade de operação do sistema tão próximo quanto possível dos seus limites de segurança. Um sistema de potência encontra-se em uma condição de emergência quando limites operacionais são violados. As mais severas e menos previsíveis violações resultam de contingências. Uma parte importante do conceito de segurança, portanto, gira em torno da habilidade do sistema em suportar efeitos de contingências [26].

Distúrbios (i.e., contingências) ocorrem freqüentemente em sistemas de potência. Exemplos comuns são súbitas mudanças na carga, falhas na geração, ou mudanças na configuração do sistema de transmissão devido à faltas e chaveamentos de linhas. Em planejamento e operação, preocupa-se com a capacidade do sistema de satisfazer a demanda das cargas na presença de distúrbios. No contexto do planejamento isto é chamado confiabilidade, e no contexto da operação é chamado segurança.

Segurança é uma condição instantânea (e variável com o tempo) que é função da robustez do sistema relativamente a iminentes distúrbios. Uma definição de segurança foi introduzida por Dy Liacco, na qual a robustez do sistema é testada, usando o modelo de fluxo de potência, em relação ao conjunto das próximas contingências. O conjunto das próximas contingências é uma coleção de distúrbios cuja seleção deve ser baseada em suas probabilidades de ocorrência e as conseqüências que delas advêm. O objetivo do controle seguro é manter afastado o risco de sobrecargas em equipamentos, tensões anormais, decaimento da freqüência, instabilidade do sistema, perda de carga, perda de geração, e o risco máximo de uma paralisação catastrófica do sistema [25].

Uma classificação formal dos níveis de segurança do sistema de potência é necessária a fim de definir as funções relevantes do *EMS*. A Figura 1.1 ilustra uma versão estendida por Stott *et al.* [27], em que as linhas com setas indicam transições involuntárias entre os níveis 1 a 5 devido à contingências.

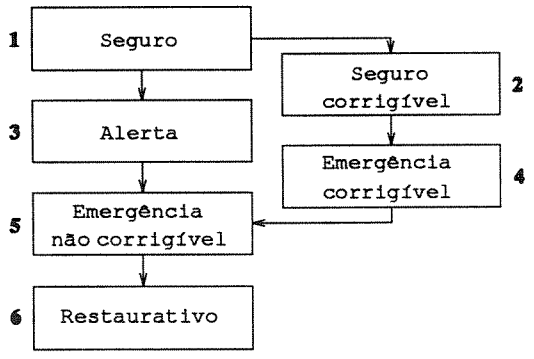


Figura 1.1: Níveis de segurança estática do sistema de potência.

A remoção de violações a partir do nível 4 geralmente exige um despacho corretivo a partir do *EMS* ou ações corretivas, conduzindo o sistema ao nível 3. Uma vez que o nível 3 tenha sido alcançado, ações oriundas do *EMS* (redespacho preventivo) devem ser efetuadas para retornar o sistema ao nível 1 ou 2, dependendo dos objetivos operacionais da segurança.

Os níveis 1 e 2 na Figura 1.1 representam a operação normal do sistema de potência. O nível 1 tem a segurança ideal: o sistema sobrevive a quaisquer das contingências relevantes sem contar com qualquer ação corretiva pós-contingência. O nível 2 é mais econômico, mas depende

de funções corretivas do *EMS* para remover com êxito violações não severas sem perda de carga, dentro de um especificado período de tempo. Observa-se que, o conceito do nível 2 aplica-se primariamente ao controle de potência ativa pós-contingência.

Os níveis de segurança estática do sistema de potência são caracterizados pelas condições operacionais, suprimento da carga, violações de limites e capacidade de correção sem perda de carga:

Nível 1: Cargas atendidas. Limites respeitados mesmo na ocorrência de contingências.

Nível 2: Cargas atendidas. Violações de limites na ocorrência de contingências podem ser corrigidas com os controles existentes.

Nível 3: Cargas atendidas. Violações de limites causadas por contingências não poderão ser corrigidas com os controles existentes.

Nível 4: Cargas atendidas. Limites operacionais violados, porém podem ser corrigidos com ações de controle.

Nível 5: Cargas atendidas. Limites de operação violados e não podem ser corrigidos por ações de controle sem que haja perda de carga.

Nível 6: Cargas não atendidas na totalidade. Limites de operações respeitados.

As pesquisas reportadas neste trabalho relacionam-se aos níveis 1 e 2 da segurança estática.

Funções de Análise de Rede

Análise de segurança e controle têm sido implementados como um "pacote" de *software* nos modernos centros de controle de energia elétrica. As principais componentes da análise de segurança *on-line* estão mostradas na Figura 1.2. As partes de monitoramento iniciam com medições em tempo-real de quantidades físicas no sistema elétrico, como fluxos em linhas, correntes, injeções de potência e magnitudes das tensões nodais, bem como o *status* dos disjuntores e das chaves. Os dados são telemedidos e, a partir das unidades terminais remotas, são enviados aos computadores do centro de controle. Dados medidos com erros grosseiros são rejeitados por filtragem através de uma simples verificação de lógica e consistência. Os dados são primeiro processados sistematicamente para se determinar a configuração do sistema (conexão dos geradores e da rede de transmissão). Então, os dados disponíveis são processados mais uma vez para se obter uma estimativa das variáveis de estado do sistema (magnitudes e ângulos das tensões nas barras). Para saber se há algum dado com erro e, em caso positivo, qual é o dado para que este seja descartado, processa-se a detecção de erros grosseiros. Análise de observabilidade, detecção e identificação de dados com erros grosseiros são partes da chamada estimação de estado generalizada.

Para determinar se um estado operacional é seguro, uma lista de contingências é necessária. A seleção de contingências emprega um esquema adaptativo para selecionar um conjunto de importantes e prováveis distúrbios. A determinação da segurança envolve essencialmente análises de fluxo de potência em regime permanente (designado de modo corriqueiro por "fluxo de carga"). Restrições são expressas em termos dos limites sobre fluxos em linhas e tensões nas barras. Portanto, para determinar a resposta do sistema às contingências, a avaliação de contingências é feita usando-se um fluxo de potência *on-line*. O fluxo de potência *on-line* utiliza o modelo do fluxo de potência convencional do próprio sistema (a partir da solução do estimador) juntamente com uma representação em tempo-real das vizinhanças do sistema, ou seja, um modelo da rede externa. Desde que as contingências são eventos futuros, uma previsão das cargas das barras é necessária [27].

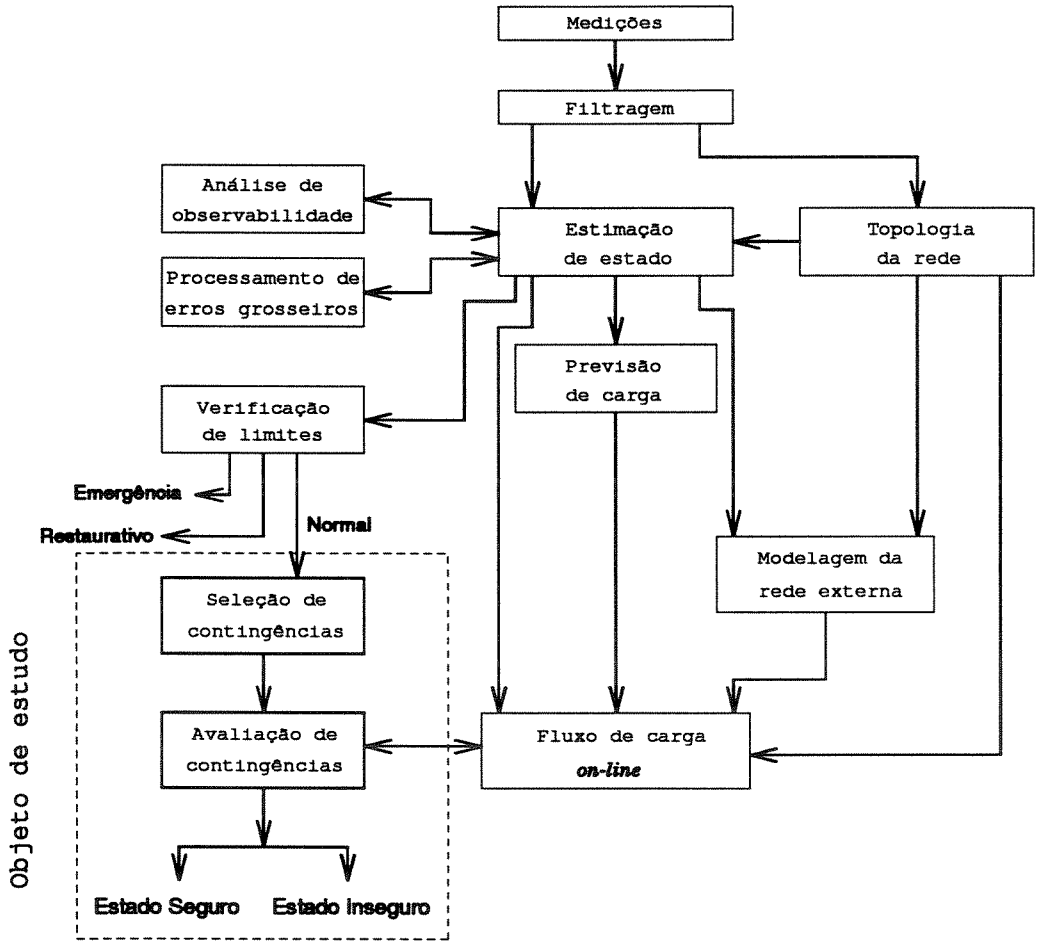


Figura 1.2: Funções de análise de rede de um Centro de Supervisão e Controle.

Na Figura 1.2 estão destacados com linha tracejada os blocos que constituem o objeto de estudo desta tese.

As principais metas ao analisar contingências continuamente num centro de controle são as seguintes [18]:

(a) Verificar se o caso-base é seguro, ou seja, se há violações, quais são, e quais são as severidades.

(b) Definir estratégias corretiva/preventiva sobre os controles da rede para tornar o sistema seguro.

Para alcançar estas metas são necessárias implementações eficientes de seleção e avaliação de contingências para monitorar sobrecargas e violações nos equipamentos e, também, programas de fluxo de potência ótimo com restrições de segurança.

Dado um caso-base e uma lista de contingências críticas, o fluxo de potência ótimo com restrições de segurança, visto aqui como uma função de análise de rede, deverá responder à seguinte questão:

"Como se deve redistribuir (ou ajustar) os controles (e.g., geração ativa) para operar o sistema de forma segura (i.e., sem violações mesmo sob distúrbios previstos) e ao mesmo tempo satisfazer a um objetivo pré-especificado?"

Em resumo, a segurança do sistema pode ser decomposta em três principais funções que são desempenhadas em um centro de controle da operação [59]:

1. Monitoramento do sistema.
2. Análise de contingências.
3. Fluxo de potência ótimo com restrições de segurança.

O monitoramento do sistema, geralmente feito através de sistemas *SCADA* (*supervisory control and data acquisition*), fornece aos operadores e aos computadores do centro de controle as condições atuais do sistema de energia elétrica. Nesta função estão englobados todos os subsistemas de telemedição, filtragem, processamento dos dados e estimação de estado da rede. É reconhecidamente a função mais importante das três.

Análise de contingências depende das informações colhidas no sistema e pré-processadas nos computadores do centro. Esta função permite operar o sistema de forma defensiva. Muitos dos problemas que ocorrem em uma rede elétrica podem causar sérios transtornos num intervalo de tempo curto tal que o operador não possa agir rápido o bastante. Por causa deste aspecto da operação dos sistemas, modernos computadores dos centros de operação são equipados com programas de análise de contingências que simulam possíveis problemas antes que eles ocorram.

Isto posto, uma parcela dos controles desempenhados pelo *EMS* é feito no modo preventivo. Por exemplo, funções associadas à segurança são baseadas em análise de contingências que efetuam a simulação de uma lista pré-definida de prováveis distúrbios: contingências de linha de transmissão, transformador, gerador e reator/capacitor em derivação (i.e., *shunt*). A análise de contingências é normalmente executada usando um caso-base obtido pela estimação de estado. O objetivo é manter a operação do sistema em condições seguras até mesmo na eventualidade da ocorrência de qualquer uma das contingências postuladas. O balanceamento entre segurança e custo é um dos maiores objetivos de um moderno centro de controle. Efetuar cálculos de contingências *on-line* exige algoritmos eficientes e *hardware* adequados.

A terceira mais importante função da segurança é o fluxo de potência ótimo com restrições de segurança. Nesta função, a análise de contingências é combinada com um fluxo de potência ótimo que busca determinar o despacho ótimo da geração, bem como outros ajustes e controles, de modo que, quando um módulo de análise de segurança é processado, nenhuma das contingências resulta em violações.

Abordagens do Problema da Análise de Segurança

Análise de contingências é a função de um *EMS* que mais consome tempo de processamento. Neste contexto, o mais difícil problema de natureza metodológica é a velocidade de solução do modelo usado. No aspecto lógico, o problema mais difícil é a solução de todas as contingências que de fato representam perigo para a segurança do sistema. Uma alternativa que parece viável é a abordagem deste problema (agora, associado à otimização [67]) através do emprego de sistemas de programação e processamento concorrentes (paralelo e distribuído).

A necessidade de solucionar problemas que exigem computação intensiva tem levado ao desenvolvimento de sistemas computacionais de alto desempenho, genericamente conhecidos como sistemas paralelo e distribuído.

O termo sistema distribuído é usado para definir uma vasta gama de sistemas computacionais, desde estações a micro-computadores interligados em rede até computadores paralelos com memória distribuída. O que eles possuem em comum são as múltiplas unidades de processamento (UCP), cada qual efetuando a sua própria tarefa e ao mesmo tempo cooperando com outras unidades através de um sistema de comunicação [57].

Um conjunto de estações de trabalho interconectadas pelo sistema *Ethernet* (cabo coaxial, velocidade de transmissão[1] da ordem de 10 [*Mbps*]), logicamente integradas como uma máquina paralela por meio de um *software-system* como o *PVM* ou o *MPI*, é um sistema típico de processamento distribuído. Sistemas como este estão se tornando cada vez mais comuns e, quando dotados de uma rede de comunicação rápida (por exemplo, um sistema de fibra óptica *FDDI* que atinge velocidades da ordem de 100 [*Mbps*]), e de processadores baseados em tecnologia *VLSI* (por exemplo, processadores *RISC*), são considerados de alto desempenho. O *SP1* e o *SP2* da

[1] [*Mbps*] é a unidade da taxa de transmissão de dados em uma rede de computadores.

IBM são exemplos de multicomputadores que se integram facilmente aos sistemas distribuídos em rede.

Tais sistemas distribuídos podem ser comparados aos computadores paralelos dedicados (e.g., *nCUBE2*) [64]. As máquinas paralelas de memória distribuída têm unidades de processamento fisicamente juntas e freqüentemente são integradas por um sistema operacional próprio.

De outro lado, existem os computadores paralelos que possuem memória global (compartilhada). Um exemplo é o computador PP desenvolvido pela Telebrás a partir de processadores 286 [43]. Já os sistemas distribuídos permitem a alteração da sua topologia e a adição de máquinas de tipos diversos.

Os sistemas de processamento distribuído, por serem economicamente viáveis e possuírem características de arquiteturas abertas (ou seja, possibilitam acompanhar a evolução e o crescimento), tendem a ser largamente utilizados nos Centros de Supervisão e Controle da indústria de energia elétrica. Uma das contribuições deste trabalho é mostrar a viabilidade do uso desses sistemas para a execução *on-line* de aplicações de computação intensiva (como os problemas de análise de contingências e fluxo de potência ótimo).

Para sistemas esparsos de equações algébricas lineares, os métodos iterativos oferecem uma alternativa atrativa relativamente aos tradicionais métodos diretos. São relativamente fáceis de programar e, para sistemas de grande porte, a velocidade de processamento de certos algoritmos iterativos pode superar a dos métodos baseados em ordenação e fatoração de matrizes (métodos diretos). Os métodos iterativos de solução de sistemas de equações algébricas lineares dividem-se em estacionários (e.g., Gauss-Seidel, Gauss-Jacobi, *SOR*, etc.) e não estacionários (e.g., gradiente conjugado, *GMRES*, etc.). Neste ramo de pesquisa, sobre análise de segurança estática foram feitas algumas contribuições quanto à aplicação do método do gradiente conjugado pré-condicionado ao problema de seleção de contingências. Os resultados foram comparados com as soluções obtidas com os métodos diretos já bastante otimizados no que se refere ao aproveitamento da esparsidade. Um outro aspecto interessante dos métodos iterativos, notadamente para aplicações a grandes sistemas, é a facilidade de desenvolver algoritmos adequados aos computadores paralelos e vetoriais [45,62].

Os desenvolvimentos na área de análise de segurança estão expandindo rapidamente; mais e mais empresas de energia estão instalando novos centros de controle e a capacidade de processamento do *hardware* está aumentando rapidamente, criando possibilidades para abordar os problemas reais de grande porte no modo *on-line*.

Estado-da-arte

A seguir são apresentados, de forma sucinta, alguns desenvolvimentos observados nos últimos anos que envolvem os tópicos de maior interesse desta pesquisa: análise de segurança e otimização, análise de contingências e fluxo de potência ótimo, sistemas de processamento paralelo e

distribuído, métodos iterativos de solução de sistemas de equações algébricas lineares, esparsidade de matrizes e vetores, e algoritmos especializados e eficientes de solução de problemas de programação linear.

- Tinney *et al.* introduzem técnicas de exploração de esparsidade de matrizes e o algoritmo de ordenação conhecido como esquema II de Tinney (ou Tinney-2) [1].
- Dommel e Tinney propõem uma abordagem geral ao problema de solução do fluxo de potência ótimo [2]. Alsaç e Stott estendem a formulação e a técnica de solução deste problema, incorporando a representação exata de restrições de contingências, visando a obtenção de um ponto de operação ótimo seguro [7].
- Stott *et al.* desenvolvem algoritmos especializados de solução de problemas de programação linear, adaptados a otimização em redes elétricas [9,10,12,13].
- Ejebe, Wollenberg, El-Abiad, Stagg, Baughman, Schweppe, Sasson e Levner propõem técnicas de seleção de contingências e introduzem índices de severidade para quantificar sobrecargas e violações [11,14,16].
- Mark Enns *et al.* propõem uma técnica de análise linearizada de contingências voltada à exploração de esparsidade [19].
- Alsaç, Stott e Tinney unificam as técnicas de análise de contingências baseadas em compensação, através de uma abordagem orientada à exploração da esparsidade de matrizes e vetores [21].
- Tinney, Brandwajn e Chan divulgam técnicas de aproveitamento de esparsidade de vetores nas operações com matrizes esparsas [22]. Eles também propõem novos métodos de refatoração parcial que atualiza os fatores LDU de uma matriz para refletir mudanças em alguns de seus elementos [24,65].
- Betancourt, Gómez e Franquelo desenvolvem heurísticas de ordenação do processo de fatoração de matrizes que incrementam a eficiência das operações com vetores esparsos bem como dos métodos de refatoração parcial [28,29].
- Mário Veiga, Monticelli, Granville e L.M.V.G. Pinto propõem a solução via decomposição de Benders para o problema de fluxo de potência ótimo com restrições de segurança que leva em conta as capacidades corretivas pós-contingência [23,26].
- K.L. Lo *et al.* comparam diferentes métodos de obtenção de *ranking* de contingências para o subproblema ativo (sobrecargas de fluxos ativos em ramos) [30]. N.D. Hatziargyriou *et al.* estudam a aplicação de técnicas de inteligência artificial ao problema de análise de contingências [51]. Meliopoulos *et al.* comparam os desempenhos de vários métodos de análise de contingências [54].
- D.P. Pinto e J.L.R. Pereira utilizam métodos de análise de contingências que usam o conceito de vizinhança e os implementa no computador paralelo *iPSC/860* (de memória distribuída) [55].

- M. Rodrigues, O. Saavedra, Monticelli, M.J. Teixeira, H.J.C.P. Pinto, Mário Veiga e M.F. McCoy mostram a viabilidade de utilização de máquinas paralelas na solução rápida do problema do fluxo de potência ótimo com contingências [33,34,56].

- Djalma M. Falcão apresenta uma visão geral das aplicações de processamento paralelo e distribuído na área de sistemas de potência em geral [57].

- S. Hao, A. Papalexopoulos e Tie-Mao-Peng apresentam uma implementação de processamento distribuído do problema de *screening* de contingências. A implementação é baseada na plataforma do sistema operacional *UNIX* e executa numa rede de estações [61].

- S.N. Talukdar e V.C. Ramesh propõem a decomposição do problema de fluxo de potência ótimo com restrições de contingências em vários pequenos subproblemas para executá-los em paralelo, segundo a técnica de multi-agentes (*Asynchronous Teams, A-Teams*). Nestes trabalhos são feitos desenvolvimentos importantes no que se refere à consideração do remanejamento pós-contingência, sobretudo em relação ao emprego de programação multi-objetivo na solução do fluxo de potência ótimo com restrições de contingências e nos aspectos conceituais deste problema [38,50,67].

- Ilija Ekmečić *et al.* resumem conceitos e sistemas de computação empregados nas pesquisas atuais. Os sistemas heterogêneos de computação abordados no artigo apresentam elevado potencial de utilização nos futuros centros de controle. São discutidos os sistemas *PVM*, *Linda*, etc., e a taxonomia de sistemas de computadores em geral [68].

Diante da exposição feita acima, percebe-se que há um sério desafio a ser vencido pelos engenheiros de potência nos próximos anos: de um lado, a análise de segurança estática impõe-se como um pesado problema matemático-computacional que possui fortes restrições de tempo de execução (i.e., prazos), requer processamento continuado[2] (i.e., a intervalos regulares de tempo) sob condições diferentes do sistema e, sua solução precisa ser confiável o bastante para que seja viável seu uso prático; de outro lado, estão disponíveis as modernas técnicas de cálculo com matrizes e vetores, e eficientes algoritmos de otimização, além dos recursos emergentes de processamento distribuído. Esta é a principal motivação para o desenvolvimento desta tese de doutorado.

Propostas deste Trabalho

Este trabalho focaliza funções de análise de segurança estática, em especial o problema de análise de contingências e técnicas de cálculo do fluxo de potência ótimo com restrições de segurança (FPORS).

A abordagem do FPORS baseia-se na linearização do modelo da rede e utiliza a técnica de programação linear (PL) dual-simplex revisado [12] e algoritmos de análise de contingências

[2] Um *software* dessa categoria deverá processar inúmeras vezes a cada 24 horas sob diferentes condições topológicas e de estado da rede elétrica.

[27]. O tratamento resume-se às variáveis de controle ativo do sistema; a geração de restrições de segurança é feita a partir de um *ranking* das contingências e se fundamenta nas metodologias de Stott (corte pela máxima violação) [43] e Benders (corte construído com multiplicadores simplex) [23].

Quanto à análise de contingências foram investigadas duas classes de métodos: (1) diretos [19,21,24] e (2) iterativos [62]. Melhorias foram introduzidas e implementadas para tornar os métodos iterativos competitivos com os métodos diretos mesmo para sistemas elétricos de médio porte.

As implementações de FPORS e as rotinas de obtenção do *ranking* foram transportadas para um ambiente de processamento distribuído (estações interligadas numa *LAN*) através do *PVM* [52], formando uma máquina paralela virtual. Foi também pesquisada a elaboração de programas tolerantes a falha, propriedade com a qual o *PVM* se destaca.

Composição deste Trabalho

O trabalho está documentado e organizado da seguinte forma: no Capítulo 2 são apresentados três programas seqüenciais de fluxo de potência ótimo com restrições de segurança elaborados segundo as metodologias de Stott e Benders; no Capítulo 3, as técnicas de análise de contingências utilizadas são apresentadas com detalhes e, além disso, é proposta e implementada uma nova técnica de seleção de contingências inspirada no fluxo de potência ótimo com restrições de segurança e na capacidade corretiva do sistema; no Capítulo 4 é apresentado o ambiente computacional de programação e processamento distribuído utilizado (configurado com o *PVM*), e nele também são detalhados e discutidos os paradigmas de programação distribuída nos quais os programas foram paralelizados; no Capítulo 5, motivado pela característica das funções *on-line* de serem processadas continuamente, é proposto um novo modelo de programação distribuída que decompõe um programa típico de análise de contingências em estágios seqüenciais/paralelos que executam em grupos de processadores, formando um *Pipeline* dotado de propriedades dinâmicas; no Capítulo 6, para ilustrar a funcionalidade e as qualidades em geral dos programas elaborados, são apresentados resultados de testes em redes elétricas padrões do *IEEE* e em redes reais de grande porte, que são partes do Sistema Elétrico Interligado do Brasil; no Capítulo 7 são ressaltados os principais aspectos do trabalho e são reunidas as mais importantes conclusões e contribuições. Ao final, são apresentados em apêndices alguns detalhes teóricos ou práticos sobre os desenvolvimentos e implementações da tese. No Apêndice A é descrito o algoritmo especializado de Stott para solução de problemas de programação linear de redes de energia elétrica. No Apêndice B, os fundamentos teóricos da decomposição de Benders são dados. O Apêndice C traz as funções das primitivas mais importantes do sistema *PVM* utilizadas neste trabalho. No Apêndice D é resolvido um pequeno exemplo numérico do fluxo de potência ótimo com restrições de segurança, aplicando-se a técnica de decomposição de Benders. Por último, no Apêndice E, métodos iterativos para resolver sistemas $[A]\mathbf{x} = \mathbf{b}$ são apresentados, discutidos e comparados com os métodos diretos de solução. Neste apêndice, o objetivo é mostrar a viabilidade de aplicação desses métodos ao processo de seleção de contingências.

Capítulo 2

FLUXO DE POTÊNCIA ÓTIMO COM RESTRIÇÕES DE SEGURANÇA

2.1 Introdução

O Fluxo de Potência Ótimo (FPO) é, essencialmente, um problema de controle no qual se procura minimizar o valor de uma função objetivo.

O cálculo do Fluxo de Potência Ótimo com Restrições de Segurança (FPORS) consiste em determinar a posição (ou o ajuste) dos controles do sistema de energia elétrica para um especificado nível de segurança, que minimiza uma função custo operacional. Nível de segurança pressupõe a introdução de análise de contingências da rede.

Matematicamente, é um problema de otimização de múltiplas variáveis com restrições de igualdade e de desigualdade, onde se busca determinar um ponto de operação factível e de mínimo custo (ou de mínimo desvio), tal que, na eventualidade da ocorrência de qualquer uma das situações postuladas (distúrbio ou contingência) através de uma lista de cenários possíveis, o estado permaneça seguro (factível).

A introdução de restrições de segurança no fluxo de potência ótimo transforma-o em um problema de otimização de grande porte, cuja dimensão, expressa em número de variáveis e de restrições, pode chegar à ordem de milhões. Diante da formidável envergadura do problema e do requisito (normalmente presente em suas aplicações) de resolvê-lo em tempo hábil a ponto de permitir tomadas de decisão, faz-se necessário abordá-lo por intermédio de formulações objetivas e inteligentes (i.e., que permitam a utilização de técnicas de relaxação, programação linear

sucessiva e decomposição) e de soluções que envolvam as novas tecnologias de processamento de alto desempenho (máquinas paralelas e sistemas de programação distribuída) [26].

Para melhor situar a abordagem dada neste trabalho ao problema de FPORS, apresenta-se a seguir uma cronologia resumida com alguns acontecimentos marcantes no desenvolvimento deste tema ao longo de três décadas [25,41].

(1962) Carpentier formula o problema de fluxo de potência ótimo;

(1968) Dommel e Tinney mostram como resolver o problema através de métodos de programação não-linear [2];

(1974) Alsaç e Stott incluem restrições de contingências e, através do método de Newton, resolvem a rede *IEEE*-30 [7];

(1979) Stott *et al.* introduzem os conceitos de base reduzida (associada ao algoritmo dual-simplex revisado) e de relaxação no tratamento das restrições funcionais e de segurança [8,9,10,12,13];

(1980) Dy Liacco desenvolve o conceito de análise de segurança [15];

(1985-87) Monticelli, Granville e Mário Veiga introduzem na formulação do FPORS a capacidade de redespacho dos controles (remanejamento) pós-contingência, as chamadas restrições de acoplamento ou de rampa [23,26];

(1987) Stott, Alsaç e Monticelli resumem as principais técnicas de análise de segurança e otimização disponíveis até então [27];

(1990-91-92) Huneault, Galiana, Balu, Bertram, Bose, Brandwajn, Cauley, Curtice, Fouad, Fink, Lauby, Wollenberg, Wrubel, Monticelli, Teixeira, Pinto *et al.* apresentam contribuições ao tema FPORS e mostram a viabilidade de sua aplicação e do emprego de ambientes de processamento paralelo [33,34,36,40];

(Atualidade) Pesquisa de novos métodos de solução, como algoritmos genéticos, teoria do meio campo e sistemas nebulosos, e a integração de abordagens distintas; estudos da viabilidade do emprego de sistemas de processamento distribuído e demonstração do seu uso em redes elétricas de grande porte [50,51,56,57,61,67].

Deve-se ressaltar que muitos outros trabalhos, alguns de grande valor, foram produzidos no decorrer do período indicado. O resumo apresentado objetiva simplesmente situar a linha desta pesquisa no contexto desse fascinante tema.

A inclusão de restrições de segurança (contingências) na formulação do fluxo de potência ótimo e a possibilidade de resolvê-lo representou um grande passo da tecnologia. Isto adicionou

considerável complexidade ao problema, originalmente difícil mesmo na sua formulação inicial, tendo em vista a diversidade de controles existentes numa rede elétrica.

O objetivo das pesquisas atuais neste tema é o de tornar o FPORS rápido e confiável o bastante para executar em tempo-real, isto é, como mais uma função de análise de rede de um moderno Centro de Supervisão e Controle. Este é de fato um desafio, dada à necessidade de uma função desse tipo cumprir prazos (da ordem de segundos ou de minutos), principalmente se executando com sistemas elétricos de grandes dimensões como os dos E.U.A. e do Brasil.

Em todas as estratégias de solução desse problema, primeiramente, as restrições críticas obtidas a partir de uma seleção de contingências (i.e., formação de um *ranking*) são adicionadas à formulação padrão. Estratégias emergentes incorporam o remanejamento das variáveis de controle dentro de limites (restrições tipo rampa) que traduzem os rápidos chaveamentos da rede para aliviar as restrições ativas impostas pela segurança. Estas novas restrições mudam a estrutura da formulação e adicionam mais complexidade à solução. Por exemplo, blocos de restrições são adicionados ao problema por cada contingência, e o remanejamento (redespacho) impõe o acoplamento entre os estados pré e pós-contingência.

A metodologia de solução do FPORS não se ajusta a qualquer outra técnica e, além disso, devido à sua complexidade computacional, os algoritmos de solução normalmente empregados são baseados em programação linear. Técnicas de decomposição (como a decomposição de Benders, conforme é mostrado neste trabalho) têm se mostrado promissoras para explorar a estrutura naturalmente disjunta entre o estado da rede intacta (caso-base) e o estado pós-contingência [40].

Na seção 2.2 são revisados alguns aspectos fundamentais do fluxo de potência ótimo na sua forma original. Pretende-se com isto relembrar conceitos chaves a respeito do tema. Posteriormente, na seção 2.3, o problema do fluxo de potência ótimo com restrições de segurança é retomado.

2.2 Formulação do Problema de Fluxo de Potência Ótimo (FPO) - Generalidades

O fluxo de potência ótimo é formulado matematicamente como um problema de otimização geral restrito, conforme (2.1).

$$\begin{cases} \text{Min} & f(\mathbf{x},\mathbf{y}) \\ \text{s.a.} & g(\mathbf{x},\mathbf{y}) = \mathbf{0} \\ & h(\mathbf{x},\mathbf{y}) \geq \mathbf{0} \\ & \mathbf{x}_{\text{mín}} \leq \mathbf{x} \leq \mathbf{x}_{\text{máx}} \end{cases} \tag{2.1}$$

Onde,

$\mathbf{x}$ é o vetor de variáveis de controle;

$\mathbf{y}$ é o vetor de variáveis dependentes;

$f(\mathbf{x},\mathbf{y})$ é a função objetivo, escalar;

$g(\mathbf{x},\mathbf{y}) = \mathbf{0}$ são as equações do fluxo de potência convencional, ocasionalmente aumentadas por algum tipo especial de restrição de igualdade;

$h(\mathbf{x},\mathbf{y}) \geq \mathbf{0}$ são os limites operacionais do sistema elétrico;

$\mathbf{x}_{\text{mín}} \leq \mathbf{x} \leq \mathbf{x}_{\text{máx}}$ representam os limites das variáveis de controle, normalmente referidas como restrições *hard* por corresponderem aos limites físicos de equipamentos e aparatos do sistema.

As variáveis de controle mais comuns podem ser classificadas de acordo com seu efeito primário sobre a potência ativa ou reativa, ou sobre ambas. Assim, surgem designações comuns na literatura, como subproblema ativo e subproblema reativo. A seguir, é apresentada a relação das variáveis de controle ($\mathbf{x}$) mais comuns nos estudos de fluxo de potência ótimo.

a.1) Subproblema ativo

geração de potência ativa;
ângulo de transformador defasador;
intercâmbio de potência ativa;
potência ativa transferida em elos de transmissão *HVDC*.

a.2) Subproblema reativo

tensões nas barras de geração ou geração de potência reativa;
taps de transformadores;

potências de capacitores ou reatores em derivação.

a.3) Subproblemas ativo e reativo

taps de transformadores com relação complexa;
geração de potências ativa e reativa;
corte de carga (MW e MVAr);
ação de chaveamento de linhas de transmissão.

As restrições operacionais, normalmente referidas como limites *soft* (porque são conhecidas de forma imprecisa, e para as quais pequenas violações podem ser toleradas), são também separadas em subproblemas ativo e reativo, conforme elas sejam mais afetadas pelos controles ativos, reativos, ou ambos. Os tipos de restrições ($h(\mathbf{x},\mathbf{y}) \geq \mathbf{0}$) que mais aparecem na literatura são dadas a seguir:

b.1) Subproblema ativo

fluxos em linhas (MW);
reserva girante ativa;
fluxos de intercâmbio (MW);
fluxos em grupos de ramos;
abertura angular das tensões das barras.

b.2) Subproblema reativo

tensões nas barras;
fluxos em linhas (MVAr);
fluxos de intercâmbio (MVAr);
fluxos em grupos de ramos.

b.3) Subproblemas ativo e reativo

limites de correntes e de fluxos de potência aparente (MVA);
fluxos em grupos de ramos (MVA).

As variáveis dependentes ($\mathbf{y} = \mathbf{y}(\mathbf{x})$) correspondem ao estado da rede, $\mathbf{y} = (\theta \mid \mathbf{V})^t$.

2.2.1 Funções Objetivos Usuais

Os quatro objetivos mais comuns serão discutidos a seguir. Outros objetivos podem ser definidos e incorporados nas implementações de FPO à medida que as necessidades de engenharia forem identificadas e tornadas claras.

Mínimo Custo de Operação

Este objetivo considera a soma dos custos dos geradores controláveis mais os custos de todas as transferências controláveis de intercâmbio. Todas as variáveis de controle do sistema são candidatas a participar na minimização deste objetivo. Se apenas as potências ativas são controladas, o processo de cálculo passa a ser chamado de Despacho Econômico de Potência Ativa (referido de modo corriqueiro como despacho). Se forem incluídas variáveis de controle sem custos diretos, como tensões e *taps*, elas participam coordenando perdas na transmissão na minimização do custo global.

Um fator crítico na minimização de custo é o modelo matemático das curvas de custos dos geradores. A maioria dos métodos requer curvas de custo convexas e, embora as curvas reais não satisfaçam este requisito, elas são aproximadas por curvas convexas. Uma maneira usual e eficiente é a aproximação da curva (por exemplo, de forma parabólica suave) por uma curva multi-segmentada, que é tratada como uma função linear por partes (vide Apêndice A).

Mínimo Desvio de um Ponto Especificado

Este objetivo é definido como a soma dos quadrados ponderados dos desvios das variáveis de controle em relação a um ponto de referência, do qual se deseja afastar o mínimo possível. Este ponto de referência será designado neste trabalho como ponto inicial. Como exemplo, no caso da geração ativa de origem hidroelétrica, a referência pode ser a melhor distribuição dos níveis de geração das usinas fornecida ao Centro de Controle pelo pessoal do planejamento da operação ligado aos assuntos hídricos. Este é um objetivo que pode ser adequado à realidade do sistema brasileiro de energia elétrica.

Mínima Perda de Potência Ativa na Transmissão

Os controles que podem ser associados a este objetivo são todos aqueles sem custo direto, exceto as gerações e intercâmbio de MW. No entanto, há controles como os defasadores e fluxos em elos de corrente contínua que não são utilizados na minimização de perdas, uma vez que estes são mais úteis ao controle da potência ativa. Portanto, a minimização de perdas é normalmente associada ao controle tensão/potência reativa e ao ajuste de *taps* de transformadores. Este é um objetivo de "ajuste fino" para a operação do sistema, que tende a diminuir a circulação de reativos e melhora o perfil de tensão. Em algumas redes elétricas, a economia de geração de potência ativa é apreciável.

Mínimo Número de Ações de Controle

A definição de mínimo número de ações é bastante livre. A intenção em geral é manejar um "pequeno" número de controles, onde "pequeno" depende das características do Centro de Controle e do sistema elétrico. Este objetivo é importante quando não é possível ou é indesejável reprogramar uma quantidade elevada de controles ao mesmo tempo. Por exemplo, quando não estão disponíveis meios automáticos para manipular vários controles simultaneamente. Este objetivo é também adequado para aliviar violações de limites operacionais em tempo-real, visando manter o sistema o mais próximo de um ponto de operação. Ao contrário do objetivo mínimo desvio, este não pode ser formulado ou resolvido com rigor matemático; tipicamente a solução será sub-ótima, porém prática em aplicações de tempo-real.

Em resumo, o FPO é um problema de otimização não-linear cujo objetivo é determinar quase todas as variáveis ajustáveis de uma rede elétrica, tais como, saídas de geradores, posições de *taps* de transformadores, posições angulares de defasadores, capacitores/reatores *shunts*, etc., para minimizar custos operacionais, perdas na transmissão, ou outros objetivos apropriados. Desde que *taps* de transformadores, ângulos de defasadores e capacitores/reatores *shunts* têm natureza discreta e, por outro lado, potências de saídas de geradores são representadas por variáveis contínuas, o FPO é um problema de programação não-linear inteiro misto. Problemas desse tipo estão sujeitos a dois inconvenientes quando de sua solução por quaisquer métodos conhecidos: (1) excessivo tempo de execução, podendo chegar à chamada explosão combinatorial; (2) acomodar-se em soluções sub-ótimas, os chamados ótimos locais [3,15].

2.2.2 Visão Geral das Soluções

Diante desses elementos complicantes e do fato de que não há ainda uma metodologia de solução que resolva esses problemas de uma maneira adequada, a política universal consiste em aproximar a formulação e o modelo da rede para resolver o FPO com métodos conhecidos e em tempos de execução aceitáveis. Portanto, deve haver um compromisso entre a formulação, o algoritmo de solução e a aplicação em engenharia (planejamento, operação, modo estudo ou modo tempo-real, etc.).

No modo tempo-real existe mais um complicante que é a necessidade de execução com prazos muito curtos e o aspecto da confiabilidade da aplicação. Outro agravante é que, fisicamente, é impossível alterar um número grande de controles ao mesmo tempo.

Na solução do FPO, quase todas as técnicas de programação matemática podem ser aplicadas. Diversas técnicas de solução do FPO têm sido desenvolvidas durante os últimos 30 anos, a saber: (a) método da iteração *lambda*, que é a base de muitas implementações atuais (em uso) do Despacho Econômico *On-line*; (b) métodos do gradiente, que apresentam convergência lenta e dificuldades para incorporar restrições de desigualdade; (c) método de Newton, que tem boa convergência, mas apresenta dificuldades no tratamento de restrições de desigualdade; (d)

método de programação linear, que se encontra bastante desenvolvido e seu uso é comum; é fácil o tratamento de restrições de desigualdade e as funções objetivos, bem como as restrições não-lineares são aproximadas por linearização; (e) método de pontos interiores, que se encontra em franco desenvolvimento e permite o tratamento de restrições de desigualdade [59].

Outras técnicas têm sido pesquisadas na solução do FPO, como a programação quadrática, as decomposições P e Q e de Benders, e as técnicas de restrições substitutas. O objetivo de todas essas pesquisas consiste na obtenção de códigos de programas computacionais que resolvam o FPO de forma confiável.

As formulações clássicas originais do FPO tiveram como pioneiros Carpentier, Dommel e Tinney [2]. Muitas dentre as pesquisas feitas no tema até então foram dirigidas a formulações similares, sem considerar os requisitos adicionais necessários às aplicações práticas *on-line*.

A abordagem do fluxo de potência ótimo segundo sua formulação geral e soluções através dos algoritmos não-lineares (por exemplo, subproblemas ativo e reativo resolvidos pelo método de Newton) foge ao escopo desta pesquisa. O enfoque dado aqui envolve formulações simplificadas (subproblema $P\theta$) e soluções por meio de algoritmos de programação linear e a inclusão de restrições lineares de segurança. Na seção 2.3 e nas suas subseções ficará clara a abordagem computacional dada a este problema nesta tese.

2.3 Formulação do Problema de Fluxo de Potência Ótimo com Restrições de Segurança (FPORS) - Generalidades

Restrições de contingências são os elementos fundamentais do controle ótimo-seguro. São elementos intrínsecos aos níveis de segurança 1 e 2 da Figura 1.1, uma combinação que representa o objetivo operacional de aparentemente todo sistema elétrico. Entretanto, somente uma pequena porção dos trabalhos sobre FPO tem considerado os problemas especiais de incluir estas restrições. A maior parte das aplicações que logrou êxito relaciona-se ao despacho de potência ativa com restrição de segurança. Nos sistemas de potência atuais, a necessidade de representar as restrições de tensão está se tornando crucial.

O número total de restrições de contingências a serem incluídas no FPO é enorme. Dependendo do tamanho do sistema de potência, cada caso de contingência (seja simples ou múltipla) pode envolver centenas ou até milhares de desigualdades. Para uma grande lista de contingências, o número total de restrições pode alcançar a casa de milhões. Felizmente, muito poucas dessas restrições estarão ativas na solução final, do contrário, o FPO com restrições de contingências seria em princípio impraticável. Técnicas de relaxação, utilizadas em todos os trabalhos dessa área (onde são provisoriamente ignoradas as restrições ativas), tornam-se particularmente efetivas.

O FPORS pode ser implementado com ou sem otimizar primeiro em relação às restrições do

caso-base (i.e., estado pré-contingência). A abordagem geral é como descrita a seguir:

a) Seleção de contingências (ou uma análise de contingências completa) é efetuada para o ponto de operação atual do sistema, de modo a identificar os N_c casos de contingências com violações ou com iminentes violações.

b) Os N_c casos selecionados, que deve ser uma pequena proporção da lista global (ou "bruta"), são incorporados ao problema de FPO; são as chamadas contingências críticas.

c) O problema de FPO é resolvido sujeito a ambos os tipos de restrições, do caso-base e pós-contingência dos casos selecionados.

d) Após a execução do passo c), com o novo ponto de operação obtido, contingências que antes apenas ameaçavam a segurança agora podem levar a severas violações. Portanto, o processo inteiro deve ser repassado voltando ao passo a) até que nenhuma violação permaneça.

As restrições pós-contingência no passo c) podem adicionar considerável esforço à solução do FPO. Muito pior, entretanto, é a necessidade de iterar o processo para eliminar as violações colaterais (i.e., inseguranças criadas a cada novo ponto obtido). É uma tarefa bastante onerosa. Em aplicações de tempo-real, as iterações podem ser dispensadas dependendo da freqüência com que o FPORS será executado.

É fácil perceber, portanto, que o fluxo de potência ótimo com restrições de segurança potencialmente representa um maciço esforço computacional, esforço que já é muito grande para análise de contingências e para o próprio FPO caso-base.

2.3.1 Formulação Clássica

A formulação original do FPO, dada em (2.1), pode ser expandida pela inclusão de restrições de contingências. Isto resulta em (2.2):

$$\begin{cases} \text{Min} \quad f(\mathbf{x}^0, \mathbf{y}^0) \\ \text{s.a.} \quad g^\nu(\mathbf{x}^\nu, \mathbf{y}^\nu) = \mathbf{0}, \qquad \text{para} \quad \nu = 0, 1, \cdots, N_c \\ \qquad\quad h^\nu(\mathbf{x}^\nu, \mathbf{y}^\nu) \geq \mathbf{0}, \qquad \text{para} \quad \nu = 0, 1, \cdots, N_c \\ \qquad\quad \mathbf{x}_{\text{mín}} \leq \mathbf{x}^\nu \leq \mathbf{x}_{\text{máx}} \end{cases} \tag{2.2}$$

Onde, o superescrito 0 representa o estado pré-contingência (caso-base) que está sendo otimizado, e o superescrito ν ($\nu > 0$) representa os estados pós-contingência para os N_c casos de

contingências previamente selecionados. As desigualdades h^ν não têm necessariamente qualquer relação com h^0, refletindo diferentes quantidades monitoradas e/ou valores limites [27].

Cada estado pós-contingência difere do estado pré-contingência como segue:

a) As restrições de igualdade g^0 mudam para g^ν de modo a refletir a saída do equipamento (contingência de linha de transmissão ou de transformador, por exemplo).

b) As variáveis de controle $\mathbf{x}^0$ respondem pelas mudanças em $\mathbf{x}^\nu$.

Referindo-se à Figura 1.1, as diferenças entre os níveis de segurança 1 e 2 são cobertas pelo item b).

O nível de segurança 1 representa a abordagem tradicional e conservativa, onde cada estado pós-contingência deve ser livre de violação sem qualquer ação corretiva pós-contingência oriunda do Centro de Controle. O conjunto de variáveis de controle para cada estado é dado pela equação (2.3).

$$\mathbf{x}^\nu = \mathbf{x}^0 + \Delta\mathbf{x}^\nu \tag{2.3}$$

Onde, $\Delta\mathbf{x}^\nu$ é a resposta espontânea dos controles automáticos do sistema elétrico. Um dos principais exemplos é a resposta do gerador (MW) baseada na inércia, conhecido por CAG.

O nível de segurança 2 oferece um custo operacional que pode ser consideravelmente mais baixo que o do nível 1. Isto é alcançado liberando-se o Centro de Controle para efetuar um remanejamento corretivo $\delta\mathbf{x}^\nu$, para remover quaisquer violações dos limites do estado contingente atual. A ação corretiva direta do Centro de Controle torna-se parte integrante do modelo do sistema de potência. Neste caso, as variáveis de controle para cada estado são dadas pela equação (2.4).

$$\mathbf{x}^\nu = \mathbf{x}^0 + \Delta\mathbf{x}^\nu + \delta\mathbf{x}^\nu \tag{2.4}$$

Neste ponto, deve-se ressaltar a existência de duas vertentes diferentes de abordagem do problema de FPORS, no que tange à capacidade de remanejamento dos controles frente à ocorrência de uma contingência:

- Fluxo de potência ótimo com restrições de segurança e sem capacidade corretiva pós-contingência (nível 1).

- Fluxo de potência ótimo com restrições de segurança e com capacidade corretiva pós-contingência (nível 2).

2.3.2 Formulações Linearizadas do FPORS

Nesta classe de formulação, os fluxos de potência nos ramos do sistema de transmissão são aproximados por um modelo linearizado:

$$P_{km} = \frac{\theta_k - \theta_m}{x_{km}} = x_{km}^{-1}\theta_{km} \quad \text{ou} \quad P_{km} = -b_{km}\theta_{km} \tag{2.5}$$

Neste caso, é fácil comprovar que o modelo matemático da rede é dado pelo sistema de equações e inequações (2.6):

$$\begin{cases} [B']\theta = \mathbf{P}_g - \mathbf{P}_d \\ \mathbf{P}_g^{\text{mín}} \leq \mathbf{P}_g \leq \mathbf{P}_g^{\text{máx}} \\ |[T]\theta| \leq \phi^{\text{lim}} \end{cases} \tag{2.6}$$

Onde,

θ é o vetor de ângulos das tensões nodais;

$\mathbf{P}_g$ e $\mathbf{P}_d$ são, respectivamente, os vetores de gerações e de demandas e $\mathbf{P}_g - \mathbf{P}_d$ resulta no vetor de injeções líquidas nas barras;

$[T]$ é a matriz incidência ramo-nó e ϕ^{lim} é o vetor de limites operacionais;

$[B']$ é uma matriz tipo admitância nodal e cujos elementos são:

$$B'_{km} = b_{km}, \quad \text{se } k \neq m$$

$$B'_{kk} = \sum_{m \in \Omega_k} -b_{km}$$

Ω_k é o conjunto das barras diretamente ligadas à barra k, e b_{km} é a susceptância do ramo $k-m$.

Por intermédio das expressões (2.6) são atendidos os seguintes requisitos básicos do modelo físico da rede elétrica:

- balanço de potência em todas as barras (que corresponde à 1ª Lei de Kirchoff);
- limites dos equipamentos de controle (por exemplo, os geradores de potência ativa);
- 2ª Lei de Kirchoff (fluxos linearizados e a equivalência potência-corrente, reatância-resistência e ângulo-tensão);
- limites operacionais dos equipamentos da rede elétrica (por exemplo, os ramos que transportam potência entre pontos da rede).

A matriz $[B']$, como definida em (2.6), é a própria matriz da meia iteração $P\theta$ (ativa) do fluxo de potência desacoplado rápido versão BX [8,20,58]. A matriz $[T]$, indicada em (2.6), aparece nesta formulação apenas para possibilitar uma representação compacta e concisa do modelo matemático. Na prática, nunca se monta tal matriz; o que se faz é expressar fluxos/aberturas angulares em função de injeções de potência:

$$[B']\theta = \mathbf{P}_g - \mathbf{P}_d \rightarrow \theta = [B']^{-1}(\mathbf{P}_g - \mathbf{P}_d)$$

$$\theta_{km} = \mathbf{e}_{km}^t\theta \rightarrow \theta_{km} = \mathbf{e}_{km}^t[B']^{-1}(\mathbf{P}_g - \mathbf{P}_d)$$

Desse modo, tomando-se θ_{km} (ou P_{km}) sobre todos os ramos da rede tem-se o mesmo resultado que considerar $[T]$, com a vantagem de poder usar os fatores triangulares de $[B']$.

Esta modelagem tem se mostrado satisfatória nas aplicações de tempo-real do FPORS, notadamente para o subproblema ativo em redes de transmissão. Resultados mais precisos e rápidos na soluções de problemas de programação linear que envolvem redes elétricas são obtidos linearizando-se as equações em torno de um ponto inicial (por exemplo, obtido de um fluxo de potência completo *AC*, ou de um fluxo *DC*, ou da saída de um estimador) e aplicando-se técnicas de programação linear sucessiva. Esta aproximação tem como conseqüência direta a transformação de um problema de otimização não-linear em um problema de programação linear (PL).

A seguir serão apresentadas e discutidas as formulações que constituem o objeto de estudo principal desta tese.

Despacho Econômico "Puro" ou FPO Caso-base

O objetivo do Despacho Econômico de um sistema de energia elétrica é o de determinar os níveis de geração que minimizam o custo operacional do sistema (medido, por exemplo, em termos do custo da geração térmica ou nuclear e de custos marginais da geração hidrelétrica) e não viola quaisquer limites operacionais como fluxos máximos em linhas de transmissão.

O Despacho Econômico "Puro" de um sistema de energia elétrica é formulado como o problema de programação linear (2.7):

$$\begin{cases} z_1 = \text{Min} \quad f(\mathbf{x}^0) \\ \text{s.a.} \qquad\quad [A]^0\mathbf{x}^0 \geq \mathbf{b}^0 \end{cases} \tag{2.7}$$

Neste problema, as variáveis $\mathbf{x}^0$ representam as decisões de operação (e.g., níveis de geração nas barras). O conjunto $[A]^0\mathbf{x}^0 \geq \mathbf{b}^0$ representa as restrições operacionais (equações de fluxo de potência e limites de fluxos). O objetivo é a minimização do custo de operação, $f(\mathbf{x}^0)$. Todo o trabalho de cálculo é feito sobre a rede intacta, isto é, sem alterações, daí o problema ser designado por FPO caso-base.

Há algum tempo já se sabe que o despacho econômico pode ser inseguro, isto é, pode não ser possível manter o sistema em um estado normal após um distúrbio grave (saída de linha ou de gerador mais importante). Isto leva ao conceito de segurança, e também à visão de que o objetivo da operação do sistema é mantê-lo em um estado normal durante períodos relativamente longos entre distúrbios e assegurar que, sob a ocorrência de um distúrbio mais severo, o sistema não se afaste do estado normal [15].

Este despacho econômico seguro é usualmente implementado pela adição de outras restrições, conhecidas como restrições de segurança, ao problema de Despacho Econômico. Estas restrições impõem limites adicionais sobre grandezas, como fluxos em ramos, para as configurações pós-distúrbio decorrentes de um dado conjunto de contingências. Em outras palavras, o despacho econômico seguro leva à implementação de ações de controle preventivo no sistema. O despacho econômico seguro será designado neste texto por fluxo de potência ótimo com restrições de segurança e a função objetivo não será necessariamente custo da geração.

Fluxo de Potência Ótimo com Restrições de Segurança

Suponha agora que se tenha uma lista de N_c possíveis contingências ou cenários, onde, cada cenário corresponde a uma situação de distúrbio, por exemplo, a saída de um ramo da rede. O conjunto de restrições operacionais para a ν-ésima configuração pós-contingência é representado como $[A]^\nu\mathbf{x}^0 \geq \mathbf{b}^\nu$, para $\nu = 1, 2, \cdots, N_c$. Desde que o ponto de operação $\mathbf{x}^0$ tem que ser factível para todas as configurações, o problema se torna como indicado em (2.8):

$$\begin{cases} z_2 = \text{Min} \quad f(\mathbf{x}^0) \\ \text{s.a.} \qquad\quad [A]^0\mathbf{x}^0 \geq \mathbf{b}^0 \\ \qquad\qquad\quad [A]^\nu\mathbf{x}^0 \geq \mathbf{b}^\nu, \qquad \text{para} \quad \nu = 1, 2, \cdots, N_c \end{cases} \tag{2.8}$$

Todavia, o despacho econômico seguro como foi definido acima é conservativo, porque ele não considera as capacidades corretivas do sistema após a contingência ter ocorrido. Estas capacidades incluem remanejamento da geração, chaveamento, rotação de sobrecarga, etc., e podem ser muito significativas para a eliminação de violações de restrições. Um exemplo pode ser encontrado em sistemas de geração predominantemente hidrelétrica, onde as taxas de respostas dos geradores asseguram que uma substancial capacidade se torne disponível num intervalo de tempo relativamente curto.

Fluxo de Potência Ótimo com Restrições de Segurança com Capacidade de Remanejamento Pós-contingência

Conforme referido atrás e ilustrado na Figura 1.1, o FPORS como definido em (2.8) assume que o ponto de operação $\mathbf{x}^0$ permanece fixo após ter ocorrido a contingência. Em outras palavras, a formulação (2.8) não leva em conta a possibilidade de ações corretivas pós-distúrbios (também referidas neste trabalho como remanejamento pós-contingência), tais como redespacho da geração a fim de eliminar violações operacionais. Essas ações corretivas podem ser introduzidas no problema de FPORS dando origem à formulação (2.9):

$$\begin{cases} z_3 = \text{Min} \quad f(\mathbf{x}^0) & \\ \text{s.a.} \qquad [A]^0\mathbf{x}^0 \geq \mathbf{b}^0 & \\ \qquad\quad [A]^\nu\mathbf{x}^\nu \geq \mathbf{b}^\nu, & \text{para } \nu = 1, 2, \cdots, N_c \\ \qquad\quad |\mathbf{x}^0 - \mathbf{x}^\nu| \leq \boldsymbol{\Delta}^\nu, & \text{para } \nu = 1, 2, \cdots, N_c \end{cases} \tag{2.9}$$

Em todas as formulações linearizadas mostradas acima, (2.7)-(2.8)-(2.9), as canalizações das variáveis de controle, $\mathbf{x}_{\text{mín}} \leq \mathbf{x}^\nu \leq \mathbf{x}_{\text{máx}}$ para $\nu = 0, 1, 2, \cdots, N_c$, foram omitidas apenas por simplicidade, contudo, elas fazem parte das formulações e são consideradas nas implementações.

O problema (2.9) permite que um novo ponto de operação $\mathbf{x}^\nu$ seja calculado após a contingência ν. Entretanto, é importante observar que há um acoplamento entre o ponto de operação do caso-base $\mathbf{x}^0$ e o ponto de operação pós-distúrbio $\mathbf{x}^\nu$; este acoplamento é dado pelas restrições da forma $|\mathbf{x}^0 - \mathbf{x}^\nu| \leq \boldsymbol{\Delta}^\nu$.

Interpretação do Vetor $\boldsymbol{\Delta}$

A consideração da capacidade de remanejamento dos controles pós-distúrbio traz em si uma interessante novidade à formulação do fluxo de potência ótimo como um problema de análise de segurança estática: a variável tempo encontra-se implícita no vetor $\boldsymbol{\Delta}$. Assim, a interpretação do vetor $\boldsymbol{\Delta}$ pode ser explicitada de forma mais clara colocando-a como uma questão decisória

de operação:

"O sistema está operando em regime normal (caso-base), no instante 0 ocorre a contingência ν, cujo tempo necessário para corrigir as anormalidades por ela causadas é t_ν. Coloca-se a questão para o operador: os controles serão capazes de mudar o ponto de operação corrente de $\mathbf{x}^0$ para $\mathbf{x}^\nu$ respeitando o intervalo t_ν ?"

A resposta será sim se o sistema estiver operando segundo o planejamento obtido através de um FPORS tal como formulado em (2.9). Obviamente, a determinação prévia de Δ^ν não está em discussão aqui e foge ao escopo deste trabalho. É um tema extremamente interessante para pesquisa, especialmente, com o advento da inteligência artificial (i.e., sistemas especialistas, redes neurais, sistemas nebulosos, etc.).

Isto posto, a formulação (2.9) pode ser reescrita de forma compacta, do seguinte modo:

$$\begin{cases} \text{Min} \quad f(\mathbf{x}^0(0)) \\ \text{s.a.} \quad \mathbf{x}^0(0) \in S_0 \\ \qquad \mathbf{x}^\nu(t_\nu) \in S_\nu, \qquad \text{para} \quad \nu = 1, 2, \cdots, N_c \end{cases} \tag{2.10}$$

Onde,

C_0 é a configuração corrente do sistema;

C_ν é a configuração resultante da contingência ν;

$\mathbf{x}^\nu(t)$ é o estado do sistema no instante t com o sistema na configuração C_ν. Mais especificamente, $\mathbf{x}^0(0)$ é o estado básico;

S_ν é o conjunto de estados normais para C_ν;

t_ν é o tempo de correção da contingência ν, ou seja, é o tempo necessário para eliminar as anormalidades causadas pela contingência;

$f(\mathbf{x}^0(0))$ é a função objetivo considerada.

Suponha que um sistema é normal, isto é, seu ponto de operação corrente, $\mathbf{x}^0(0)$, está dentro do conjunto de estados normais, S_0. Se a ν-ésima contingência ocorrer, levando o estado a uma situação anormal $\mathbf{x}^\nu(0_+)$ (fora de S_ν), a configuração do sistema imediatamente mudaria e algum tempo t_ν seria consumido para restabelecer o estado normal $\mathbf{x}^\nu(t_\nu)$. O tempo de correção t_ν existe porque as ações de controle não são instantâneas (i.e., tendem a ser *rate-limited*). A Figura 2.1 representa graficamente este efeito.

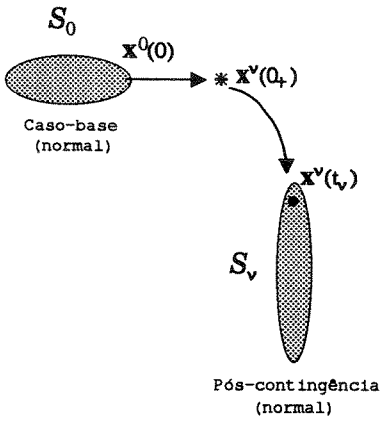


Figura 2.1: Representação gráfica da mudança de estado do sistema quando uma contingência ocorre.

Ligações entre as Formulações Linearizadas

As restrições de acoplamento também servem como uma "ponte" entre o Despacho Econômico "Puro" (2.7) e o Despacho Econômico Seguro (2.8). É possível ver que, se $\boldsymbol{\Delta} = \mathbf{0}$ (i.e., nenhuma ação corretiva é permitida), o problema (2.9) torna-se idêntico ao (2.8). Por outro lado, se $\boldsymbol{\Delta} \rightarrow \infty$ (i.e., toda flexibilidade de remanejamento é permitida no período), a formulação (2.9) recai sobre o Despacho Econômico "Puro" (2.7), já que $\mathbf{x}^\nu$, $\nu = 1, 2, \cdots, N_c$, tornam-se independentes do ponto $\mathbf{x}^0$. (Para $\boldsymbol{\Delta} \rightarrow \infty$ vide também a seção 3.4 do capítulo seguinte.)

Verifica-se, também, quanto às funções objetivos dos problemas (2.7)-(2.8)-(2.9) que seus valores z_1, z_2 e z_3 guardam a seguinte relação:

$$\text{Despacho "puro"} \rightarrow z_1 \leq z_3 \leq z_2 \leftarrow \text{FPORS sem remanejamento}$$

A Figura 2.2 compara através de um exemplo simples as três formulações: **A** representa a solução insegura, a não ser que $\boldsymbol{\Delta}$ seja ∞; **B** é a solução segura porém conservativa; **C** é a solução tão segura quanto **B** e mais "barata" porque leva em conta o remanejamento; **B1** e **C1** ilustram situações pós-contingência.

Na seção 2.4 serão apresentados os principais aspectos das implementações computacionais do FPORS desenvolvidas nesta pesquisa restringindo-se aos programas seqüenciais. No Capítulo 3 serão apresentados detalhes dos algoritmos de análise/seleção de contingências, como seguimento natural deste capítulo. No Capítulo 4 serão apresentadas e discutidas as implementações concorrentes em sistemas computacionais distribuídos.

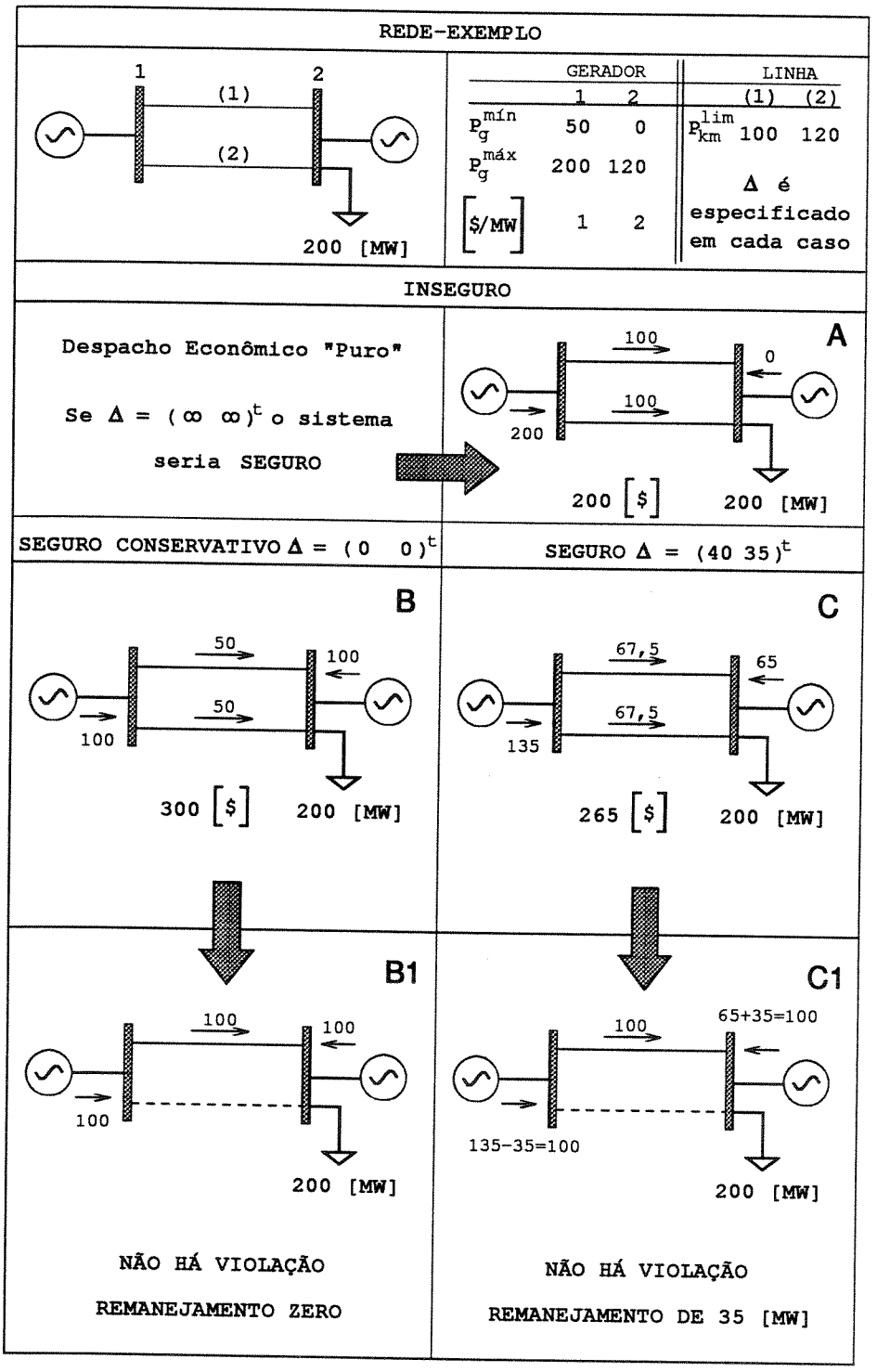


Figura 2.2: Comparações numéricas entre as formulações (2.7), (2.8) e (2.9).

2.4 Técnicas de Solução e Implementações Computacionais do FPORS

No contexto das formulações linearizadas apresentadas anteriormente,expressões (2.7)-(2.8)-(2.9), a solução do fluxo de potência ótimo com restrições de segurança requer três componentes essenciais:

(1) um algoritmo de solução de PL eficiente e adequado a redes elétricas;

(2) um fluxo de potência linearizado (fluxo *DC*) ou um fluxo de potência completo, de preferência uma versão do desacoplado rápido ou meia iteração (ativa);

(3) análise de contingências, seja através de uma avaliação completa ou de algoritmos de seleção de contingências, preferencialmente rápidos e orientados à exploração da esparsidade de matrizes e vetores.

O algoritmo de solução de PL utilizado neste trabalho é um algoritmo especializado para problemas de otimização de redes elétricas desenvolvido por Stott, Marinho e Hobson no final da década de '70 [12]. A eficiência deste algoritmo está consagrada e sua utilização consolidada entre os pesquisadores da área de sistemas de potência.

Os algoritmos de seleção de contingências, em especial para o subproblema $P\theta$, baseiam-se no fluxo de potência desacoplado rápido e nas técnicas de compensação com esparsidade de matrizes e vetores [21]. Fazem parte da família dos métodos diretos de análise de contingências. Esta parte do trabalho será estudada no capítulo subseqüente. Técnicas iterativas para solucionar sistemas de equações algébricas lineares foram também estudadas como alternativa aos procedimentos de solução direta, e estão explanadas no Apêndice E.

Com base nesses componentes[1] foram confeccionados os programas digitais de cálculo do fluxo de potência ótimo com restrições de segurança, especificamente para o subproblema ativo. Essencialmente, esses programas são capazes de resolver problemas de sobrecargas em redes elétricas pela ação dos controles associados à potência ativa.

São três os programas de fluxo de potência ótimo com restrições de segurança implementados:

I- FPORS segundo o algoritmo de relaxação originalmente proposto em [9,10,12,13] e, posteriormente, estudado em [43]. Esta pesquisa utiliza relaxação e construção do corte (restrição) a partir da máxima violação, não considera remanejamento pós-contingência e, por isso,

[1] Nas implementações não foram utilizados "pacotes" comerciais, nem na fase de PL (e.g., *MINOS, LINDO,* etc.) e nem nos cálculos de fluxo de potência (e.g., *ANAREDE* e outros). Todas as rotinas foram desenvolvidas no LSEE-UNICAMP ao longo dos anos e no decorrer das pesquisas.

fornece uma solução conservativa. Doravante, para facilitar a referência no texto, este programa será designado por FPORS-I.

II- FPORS com capacidade de remanejamento pós-contingência, implementado como uma extensão das idéias contidas no programa FPORS-I. A exemplo do primeiro, este programa utiliza relaxação e construção do corte a partir da máxima violação. Para cada contingência crítica, o programa pode ter que resolver dois problemas de programação linear: (a) no primeiro PL procura-se resolvê-lo redefinindo os limites dos geradores em função da faixa Δ e da solução corrente (do PL do caso-base ou de investimento) – é o PL de operação; (b) se houver violação remanescente, após o PL de operação ter sido computado, esta violação é introduzida no PL do caso-base sob a forma de restrição, cuja solução constituir-se-á em nova proposta de investimento. Este programa será designado por FPORS-II.

III- FPORS com capacidade de remanejamento pós-contingência, implementado segundo a técnica de decomposição de Benders. Consiste na abordagem sugerida em [23,26] e se baseia na construção da restrição a partir do vetor de multiplicadores simplex correspondentes à solução de subproblemas de operação. Daqui em diante, este programa será referido como FPORS-III.

Antes de entrar propriamente na descrição dos programas digitais será feita uma revisão do algoritmo de solução de PL empregado neste trabalho. Para obter mais detalhes é sugerida a leitura do Apêndice A. Este algoritmo se constitui no bloco de otimização dos programas implementados nesta tese.

Segundo Stott, a principal motivação para pesquisar e propor um algoritmo especializado de PL são as deficiências dos métodos de programação não-linear, os quais globalmente têm recebido muito mais esforço de desenvolvimento. Essas deficiências incluem não-confiabilidade ou convergência lenta, a necessidade de um ponto de partida factível (viável), algoritmos complicados e/ou de difícil sintonização, sofisticadas manipulações de matrizes esparsas, dificuldades no reconhecimento de "infactibilidades" e na análise de sensibilidade de soluções ótimas. Todos estes inconvenientes estão ausentes no algoritmo de solução de PL proposto.

2.4.1 Algoritmo Especializado de Stott: Dual-Simplex Revisado

O algoritmo requer a linearização do modelo matemático do fluxo de potência da rede, para tanto, inicia-se o processo construindo-se um modelo incremental em torno de um ponto inicial ($\mathbf{P}^{\text{inicial}}$). Um fluxo *DC* tem se mostrado adequado, embora, se maior precisão for desejada nos cálculos, a solução do PL pode ser iterada com um fluxo de potência *AC*.

Um modelo incremental é adotado:

$$\begin{aligned}\mathbf{\Delta P} &= \mathbf{P} - \mathbf{P}^{\text{inicial}} \\ \mathbf{\Delta P}^{\text{máx}} &= \mathbf{P}^{\text{máx}} - \mathbf{P}^{\text{inicial}} \\ \mathbf{\Delta P}^{\text{mín}} &= \mathbf{P}^{\text{mín}} - \mathbf{P}^{\text{inicial}}\end{aligned}$$

As variáveis do problema são as mudanças nas unidades de geração $\mathbf{\Delta P} = (\Delta P_1 \cdots \Delta P_n)^t$.

$$\begin{cases} \text{Min} \quad f(\mathbf{\Delta P}) = \mathbf{c}\mathbf{\Delta P} \\ \text{s.a.} \quad \beta^t \mathbf{\Delta P} = 0 \\ \qquad \mathbf{L}^{\text{mín}} \leq [A]\mathbf{\Delta P} \leq \mathbf{L}^{\text{máx}} \end{cases} \tag{2.11}$$

Onde,

$\mathbf{c}$ é o vetor de custos (incrementais) de dimensão $(1 \times n)$;

$\mathbf{\Delta P}$ é o vetor das variáveis de decisão, cuja dimensão é $(n \times 1)$;

β é o vetor de fatores de perdas na transmissão e $\beta^t \mathbf{\Delta P} = 0$ é a equação de balanço de potência ativa (reporte-se ao Apêndice A para obter mais detalhes);

$\mathbf{L}^{\text{mín}}$ e $\mathbf{L}^{\text{máx}}$ são os vetores de limites das restrições e $[A]$ é uma matriz de coeficientes das restrições lineares. (As linhas de $[A]$ são compostas de vetores dos tipos $\mathbf{A}_j^t$ e $\mathbf{A}_{km}^t$.)

As restrições de desigualdade $\mathbf{L}^{\text{mín}} \leq [A]\mathbf{\Delta P} \leq \mathbf{L}^{\text{máx}}$ são de três tipos diferentes:

- Restrições de geradores

$$\mathbf{A}_j^t = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & \cdots & 1 & \cdots & 0 \\ \hline \end{array} \quad (j)$$

- Restrições de fluxos em ramos

$$\mathbf{A}_{km}^t = -b_{km}\mathbf{e}_{km}^t[B']^{-1} \tag{2.12}$$

- Restrições de segurança (serão posteriormente analisadas)

  Para a contingência do ramo $i-j$, o fluxo incremental contingente no ramo $k-m$ pode ser calculado em termos do estado angular pré-contingência, $\Delta\theta^0_{ij}$ e $\Delta\theta^0_{km}$. Assim, é possível expressar Δf_{km} como função de $\mathbf{\Delta P}$. Isto está mostrado com mais detalhes no Capítulo 3, inclusive estendendo-se o equacionamento para contingências múltiplas de ramos.

Neste algoritmo, o método dual-simplex revisado de solução de PL é aplicado ao problema primal, usando conceitos e critérios do dual, base reduzida e técnicas de relaxação. O processo parte de um estado operacional do sistema de potência contendo ramos cujos limites de fluxos estão violados. Esta solução inicial é otimista (dual factível). Por exemplo, se a função objetivo for o custo de operação (minimizar), a solução inicial pode ser obtida colocando-se todos os geradores controláveis no limite inferior e aumentando-se a geração em ordem crescente de custo incremental até satisfazer o balanço de potência. Esta solução inicial, que atende apenas aos limites de geração, é conhecida como base inicial. Até este ponto, as restrições de fluxos nos ramos estão relaxadas. Em seguida, através de um monitor de fluxos (baseado no fluxo de potência *DC*) verifica-se se seus limites estão respeitados. Os limites de fluxos em ramos que se encontram violados são traduzidos na forma de restrições lineares que são impostas uma a uma ao processo PL. A cada nova restrição que entra na base tem-se um novo ponto de operação cuja otimalidade deve ser testada por critérios próprios. A obtenção de uma nova solução sempre é restrita a não ter violações nas variáveis de controle (restrições de geradores). Para cada novo nível de geração que satisfaz os critérios de otimalidade tem-se um correspondente estado que permitirá a verificação de novas violações de ramos (monitoramento de fluxos). Como resultado deste processo, a solução é gradativamente melhorada pela adição seqüencial de restrições de ramos e de geradores, até atingir a viabilidade de todas elas. A Figura 2.3 ilustra, por meio de um diagrama de blocos, os passos mais importantes deste algoritmo. O bloco D, denominado "Núcleo do processo PL dual-simplex revisado", é de fato a parte engenhosa do algoritmo e será detalhado no Apêndice A. Na figura omite-se a obtenção da base inicial, que é uma tarefa relativamente simples e depende do tipo de função objetivo a minimizar.

Durante o processo iterativo, somente um pequeno subconjunto das restrições esparsas são tratadas. Restrições não-esparsas (de ramos) são criadas apenas para os poucos fluxos cujos limites são executados (impostos). O método é altamente eficiente uma vez que o número de restrições de ramos ativas na solução final não é excessivo.

Embora seja fácil de implementar, o método difere de todas as versões simplex de resolução de PL que aparecem nos livros-textos. As variáveis normais de sistemas de potência com limites em ambos os lados são utilizadas por toda parte no processo de solução, não é requerida fase I ou processos similares e não são exigidas condições de não-negatividade [3,35]. O método comporta facilmente a introdução de variáveis artificiais, pela simples adição de gerações fictícias nas barras, conforme foi feita neste trabalho.

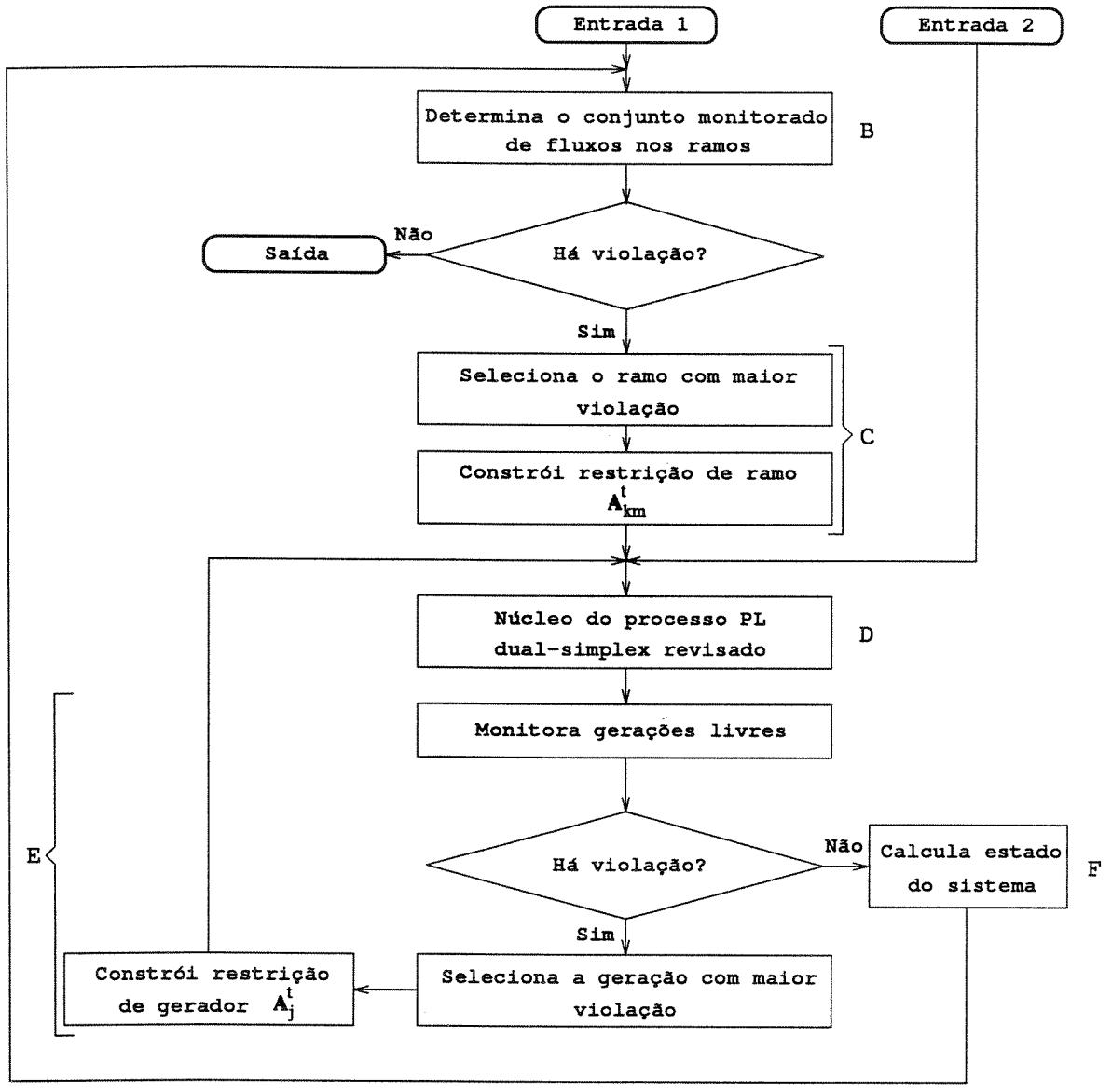


Figura 2.3: Diagrama de blocos do algoritmo de solução especializado de Stott.

O Vetor λ de Custos Marginais

O processo de solução de PL de Stott fornece diretamente o vetor de variáveis duais, λ. Cada elemento λ_j é a redução incremental na função objetivo se o limite da j-ésima restrição ativa for relaxado. Este vetor é calculado a cada iteração do método pós-multiplicando o vetor original de custos das gerações pela matriz base inversa, conforme está descrito no Apêndice A. λ é indispensável na construção de restrições de segurança na técnica de decomposição de Benders aplicada em uma das implementações estudadas nesta tese.

Agora, passando-se à descrição dos programas FPORS, chama-se a atenção para os seguintes fatos:

- o algoritmo de solução de PL ilustrado na Figura 2.3 pode ser visto como um bloco de duas entradas e uma só saída, como mostra a Figura 2.4. A entrada 2 é utilizada somente quando se deseja introduzir uma restrição de contingência no PL; a entrada 1 em caso contrário, isto é, quando se otimiza a rede intacta;

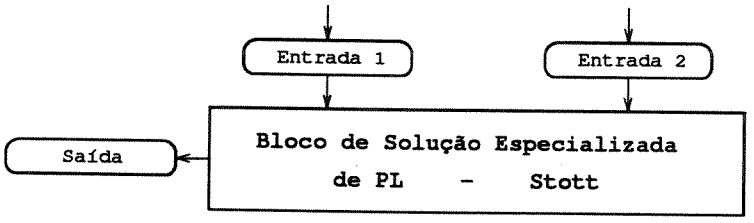


Figura 2.4: Algoritmo de solução especializado de Stott visto como uma "caixa-preta": duas entradas e uma saída.

- a Figura 2.5 mostra através de um diagrama de blocos as etapas preliminares dos programas implementados nesta tese. Os blocos indicados como "Caso-base" referem-se, na verdade, à solução de um FPO com a rede sem alterações (o que corresponde a utilizar o bloco de otimização - Figuras 2.3 e 2.4 - pela entrada 1). Dois tipos de funções objetivos são disponíveis: (1) custo de operação; e (2) desvio quadrático de um ponto especificado. Esta figura foi apresentada à parte, simplesmente, para evitar sua repetição nos diagramas ilustrados nas Figuras 2.6, 2.7 e 2.9.

2.4.2 Relaxação e Construção do Corte a partir da Máxima Violação (FPORS-I)

O programa FPORS-I resolve problemas formulados segundo (2.8), cuja modelagem incremental foi dada em (2.11). A capacidade de remanejamento pós-contingência é nula, o que

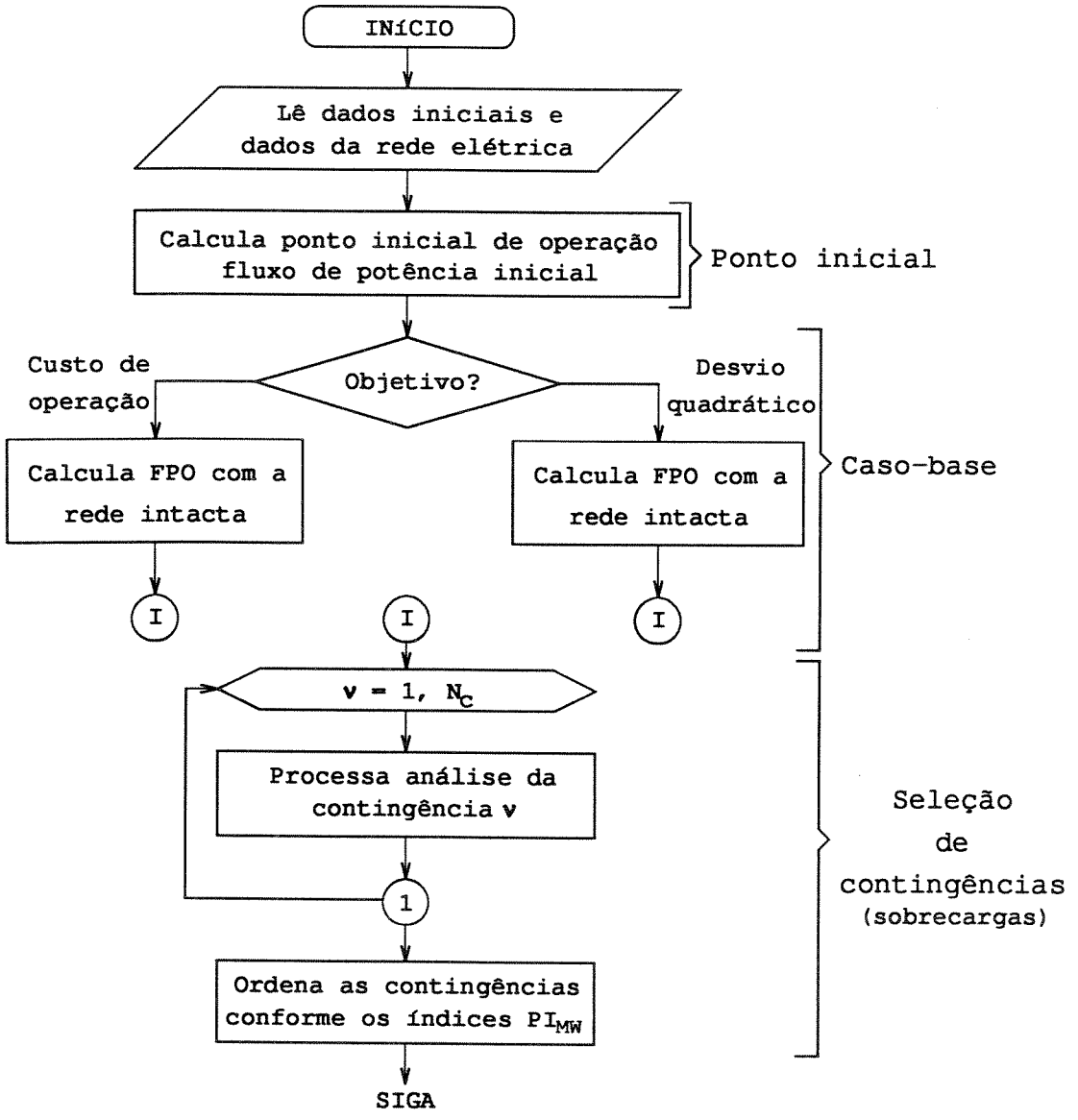


Figura 2.5: Etapas preliminares de qualquer programa de FPORS: (1) ponto inicial; (2) caso-base; e (3) obtenção do *ranking* das contingências.

corresponde ao problema/solução sugeridos por Stott *et al.* em [12].

Este programa combina programação linear dual e técnicas de relaxação. O modelo (2.11) pode ser visto como um problema com dois níveis de relaxação:

- um nível inferior, onde são consideradas apenas as restrições de geradores e de fluxos em ramos. Resolvê-lo corresponde a solucionar o FPO caso-base, isto é, sem considerar as restrições de segurança. A partir dessa solução obtém-se um *ranking* das contingências, conforme ilustra a Figura 2.5;
- no nível superior de relaxação, de posse da solução inicial (nível inferior) e do *ranking*, as contingências são monitoradas com respeito às violações nos estados pós-contingentes. Se não houver casos que levem a violação pós-contingência, então a solução será ótima-segura. Em caso contrário, é selecionada a violação mais severa e, a partir dela, constrói-se uma restrição em termos das variáveis de controle, que é introduzida na base corrente. Tal como no nível anterior, a solução é melhorada gradativamente pela incorporação de restrições de segurança. A Figura 2.6 ilustra, com diagrama de blocos, o trabalho executado neste nível.

Alguns aspectos devem ser ressaltados sobre o fluxograma da Figura 2.6. A primeira observação é que, a análise da contingência ν, a atualização do conjunto crítico (conjunto de ramos com fluxos pós-contingentes próximos dos seus respectivos limites - ramos candidatos à violação ou violados) e a obtenção da restrição de contingência ($\mathbf{A}^t_{km}$) são processos matematicamente interligados (vide seção 3.3, Capítulo 3). Isto significa que alguns dos vetores e escalares calculados durante a análise da contingência ν servem também aos cálculos subseqüentes. Durante a confecção das implementações esta propriedade foi aproveitada, resultando em melhorias no desempenho do processamento. A segunda observação importante é a presença do bloco de otimização (com o uso da entrada 2), detalhado atrás na Figura 2.3. Outro ponto relevante é o emprego do critério de marcas, originalmente proposto em [56], usado para administrar o exaustivo processo iterativo mencionado no passos **c)** e **d)** da abordagem geral do FPORS, da seção 2.3. Ainda neste capítulo, este critério será explicado em detalhes.

2.4.3 Relaxação e Construção do Corte a partir da Máxima Violação: Extensão ao Caso com Capacidade de Remanejamento Pós-contingência (FPORS-II)

O programa FPORS-II resolve o fluxo de potência ótimo com restrições de segurança e com capacidade de remanejamento pós-contingência, formulado segundo (2.9). A Figura 2.7 ilustra os passos principais deste programa, após executadas as etapas preliminares (Figura 2.5).

A presença de restrições do tipo rampa, $|\mathbf{x}^0 - \mathbf{x}^\nu| \leq \Delta^\nu$, requer que a implementação possua dois blocos de otimização (PL) a cada ciclo de iteração.

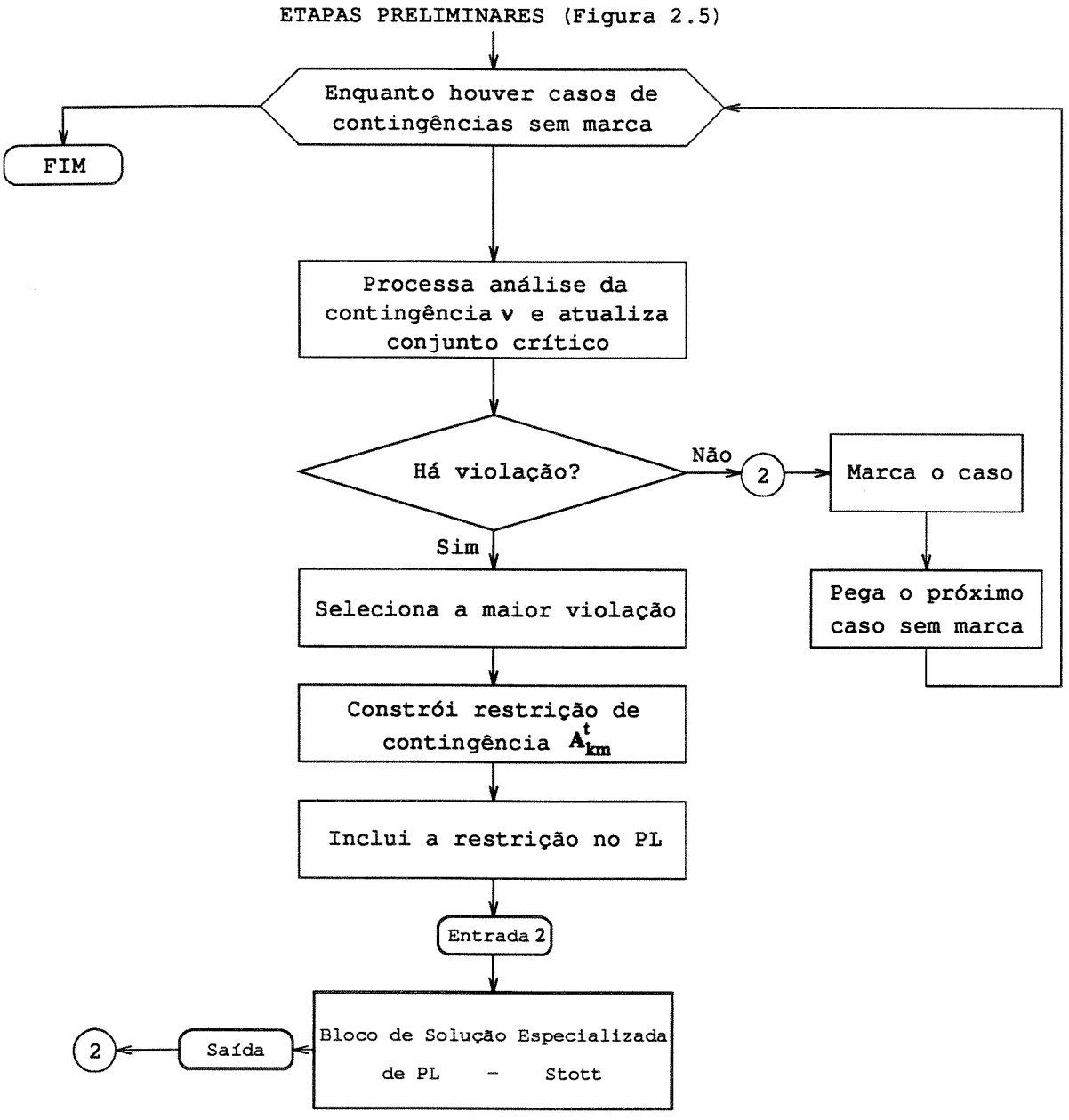


Figura 2.6: Diagrama de blocos do fluxo de potência ótimo com restrições de segurança sem capacidade de remanejamento pós-contingência (FPORS-I).

Referindo-se ao programa FPORS-II e ao diagrama da Figura 2.7, faz-se a seguinte explicação. No primeiro bloco, conhecida uma solução provisória[2] do problema ($\mathbf{\Delta P}^{0,*}$) e definida a faixa $\mathbf{\Delta}$ para as variáveis de controle (agora na forma incremental), tem-se o PL de operação no qual os limites das variáveis de decisão são assim determinados:

$$\text{Máx}\{\mathbf{\Delta P}^{\text{mín}}, -\delta^{\nu} + \mathbf{\Delta P}^{0,*}\} \leq \mathbf{\Delta P}^{\nu} \leq \text{Mín}\{\mathbf{\Delta P}^{\text{máx}}, \delta^{\nu} + \mathbf{\Delta P}^{0,*}\} \tag{2.13}$$

Onde, δ^{ν} equivale ao vetor $\mathbf{\Delta}^{\nu}$ expresso na forma incremental.

Ao resolver esse PL, de posse da restrição correspondente à máxima violação, procura-se com o remanejamento disponível eliminar a sobrecarga relativa à contingência analisada. A função objetivo do PL de operação é o objetivo original do problema. Caso o remanejamento seja suficiente, o ponto $\mathbf{\Delta P}^{0,*}$ é uma solução válida para o caso-base (investimento) e nova contingência será processada. Em caso contrário, há violação remanescente e a restrição deve ser incluída no PL cujos limites das variáveis de controle são os originais (PL de investimento ou PL do caso-base – ou simplesmente caso-base). Passa-se assim ao segundo bloco de otimização.

O primeiro bloco pode ser visto como uma rotina de verificação da factibilidade da solução corrente considerando-se a faixa $\mathbf{\Delta}$. O segundo bloco é decisório no que se refere ao investimento e só será executado se, com a faixa de remanejamento, não for possível corrigir a sobrecarga.

Nota-se, pela comparação das Figuras 2.6 e 2.7, as semelhanças entre os dois programas. A principal diferença é a existência do primeiro bloco (PL de operação) no programa FPORS-II, justamente com a função de tratar as restrições de acoplamento. Neste caso, o bloco otimizador que trata o PL de operação tem, regra geral, estrutura muito parecida com o diagrama da Figura 2.3. A diferença praticamente se resume no teste de elegibilidade para a escolha da geração a entrar na base: se não houver geração elegível, relaxa-se a restrição de contingência e, com isto, se tem a medida de quanto deve ser o novo limite da restrição a ser incluída no segundo bloco (PL de investimento ou PL do caso-base).

Ressalta-se que a implementação conforme a Figura 2.7 traz em si uma idéia inteligente e prática. Com os mesmos conceitos de relaxação e construção da restrição pela máxima violação empregados no programa FPORS-I (Figura 2.6) foi possível confeccionar um programa que trata restrições de acoplamento, em princípio distintas das restrições da forma $\mathbf{A}^t\mathbf{\Delta P}$ de Stott, sem usar decomposição. Conforme está mostrado à frente, no Capítulo 6, esta implementação tem excelente desempenho computacional.

[2] No início, a solução dita provisória é a própria saída do FPO caso-base (Figura 2.5). A cada ciclo que se completa tem-se um novo vetor $\mathbf{\Delta P}^{0,*}$.

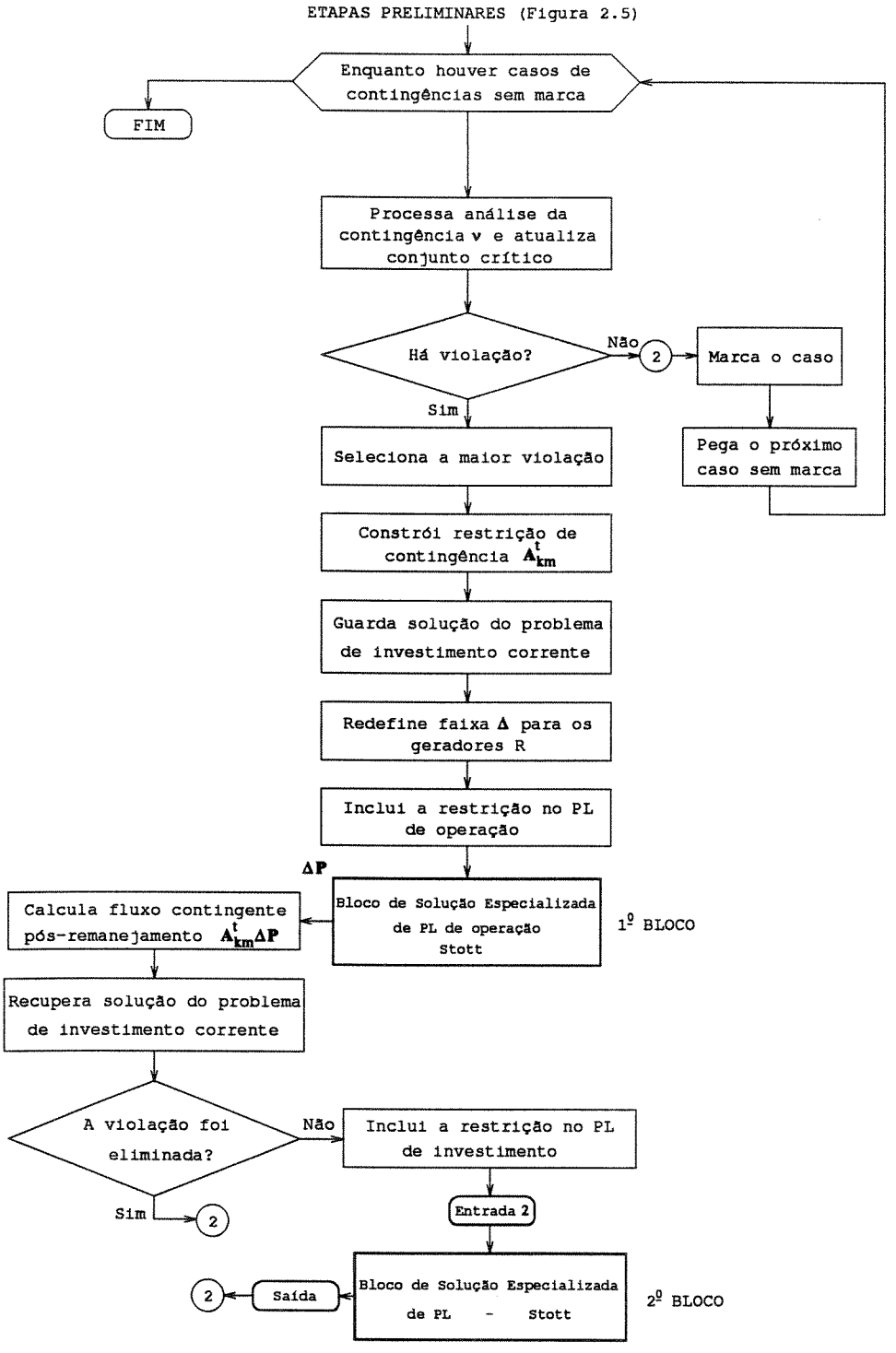


Figura 2.7: Diagrama de blocos do fluxo de potência ótimo com restrições de segurança com capacidade de remanejamento pós-contingência (FPORS-II).

2.4.4 Decomposição de Benders Aplicada ao Problema de FPORS com Capacidade de Remanejamento Pós-contingência (FPORS-III)

Nesta implementação aplica-se a decomposição de Benders, desenvolvida por Geoffrion conforme [6]. Esta técnica é adequada à solução de problemas de otimização do tipo de (2.9), onde há variáveis normais $\mathbf{x}^{\nu}$ com $\nu > 0$ e variáveis complicantes $\mathbf{x}^{0}$, e estas variáveis encontram-se acopladas por intermédio de pelo menos uma das restrições. No Apêndice B é feita uma breve introdução aos fundamentos matemáticos desta decomposição.

Em particular, o problema (2.9) é decomposto segundo a metodologia explicada em [23,26]: (1) problema de investimento ou do caso-base reescrito segundo o modelo incremental dado pelas expressões (2.14) e (2.16); (2) subproblemas de operação na situação pós-contingência, ou seja, com a rede alterada em função da respectiva contingência. Na forma incremental tem-se o subproblema para a contingência ν como indica (2.15).

Problema de investimento inicial - FPO caso-base

$$\begin{cases} \text{Min} & \mathbf{c}\Delta\mathbf{P}^{0} \\ \text{s.a.} & \beta^{t}\Delta\mathbf{P}^{0} = 0 \\ & \mathbf{L}^{\text{mín}} \leq [A]^{0}\Delta\mathbf{P}^{0} \leq \mathbf{L}^{\text{máx}} \end{cases} \tag{2.14}$$

Subproblema de operação - FPO com a rede alterada pela contingência ν

$$\begin{cases} w^{\nu} = \text{Min} & \mathbf{d}\Delta\mathbf{P}^{\nu} \\ \text{s.a.} & \beta^{t}\Delta\mathbf{P}^{\nu} = 0 \\ & \mathbf{L}^{\text{mín}} \leq [A]^{\nu}\Delta\mathbf{P}^{\nu} \leq \mathbf{L}^{\text{máx}} \\ & |\Delta\mathbf{P}^{0,*} - \Delta\mathbf{P}^{\nu}| \leq \delta^{\nu} \end{cases} \tag{2.15}$$

Na expressão (2.15), o vetor δ^{ν} está associado ao vetor Δ^{ν} original aqui transformado para o modelo incremental e $\mathbf{d}$ é o vetor de custos incrementais, $(1 \times n)$, que está definido logo à frente.

A metodologia de solução empregada nesta implementação trata o problema como um processo de decisão de dois estágios: (1) no primeiro estágio, obtém-se um ponto de operação $\Delta\mathbf{P}^{0,*}$ resolvendo-se (2.14) na iteração inicial e, nas iterações seguintes, (2.16) (nota-se que (2.16) já apresenta os cortes de Benders); (2) no segundo estágio, dado o ponto $\Delta\mathbf{P}^{0,*}$, são obtidos os

pontos $\Delta\mathbf{P}^{\nu,*}$ que atendem às restrições de balanço, operacionais e de acoplamento nos estados pós-contingência, $\nu = 1, 2, \cdots, N_c$. Enquanto no primeiro estágio a função objetivo é a escolhida para o problema original (custo de operação ou desvio quadrático), no segundo estágio, os PLs de operação são resolvidos minimizando-se a "infactibilidade"[3].

Problema de investimento com os cortes de Benders

$$\begin{cases} \text{Min} \quad \mathbf{c}\Delta\mathbf{P}^0 \\ \text{s.a.} \quad \beta^t\Delta\mathbf{P}^0 = 0 \\ \qquad \mathbf{L}^{\text{mín}} \leq [A]^0\Delta\mathbf{P}^0 \leq \mathbf{L}^{\text{máx}} \\ \qquad w^\nu(\Delta\mathbf{P}^{0,*}) \leq 0, \qquad \text{para } \nu = 1, 2, \cdots, N_c \end{cases} \tag{2.16}$$

A expressão $w^\nu(\Delta\mathbf{P}^{0,*}) \leq 0$, que aparece no problema (2.16), representa o que se chama de corte de Benders, cuja determinação será discutida logo à frente.

A Figura 2.8 mostra o esquema iterativo do algoritmo de decomposição de Benders tal como empregado neste trabalho para resolver as expressões (2.14)-(2.15)-(2.16).

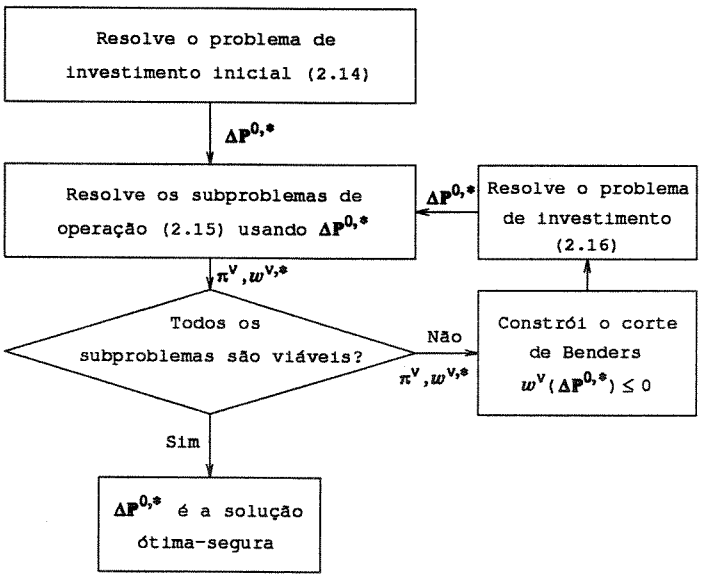


Figura 2.8: Algoritmo de decomposição de Benders.

O diagrama de blocos ilustrado na Figura 2.9 permite uma visão detalhada dos passos mais significativos da implementação do FPORS de acordo com a técnica de decomposição de Benders.

[3]O termo "infactibilidade" é usado aqui como uma medida do quanto faltou em geração ou o quanto será preciso rejeitar da carga numa barra para viabilizar a solução do PL.

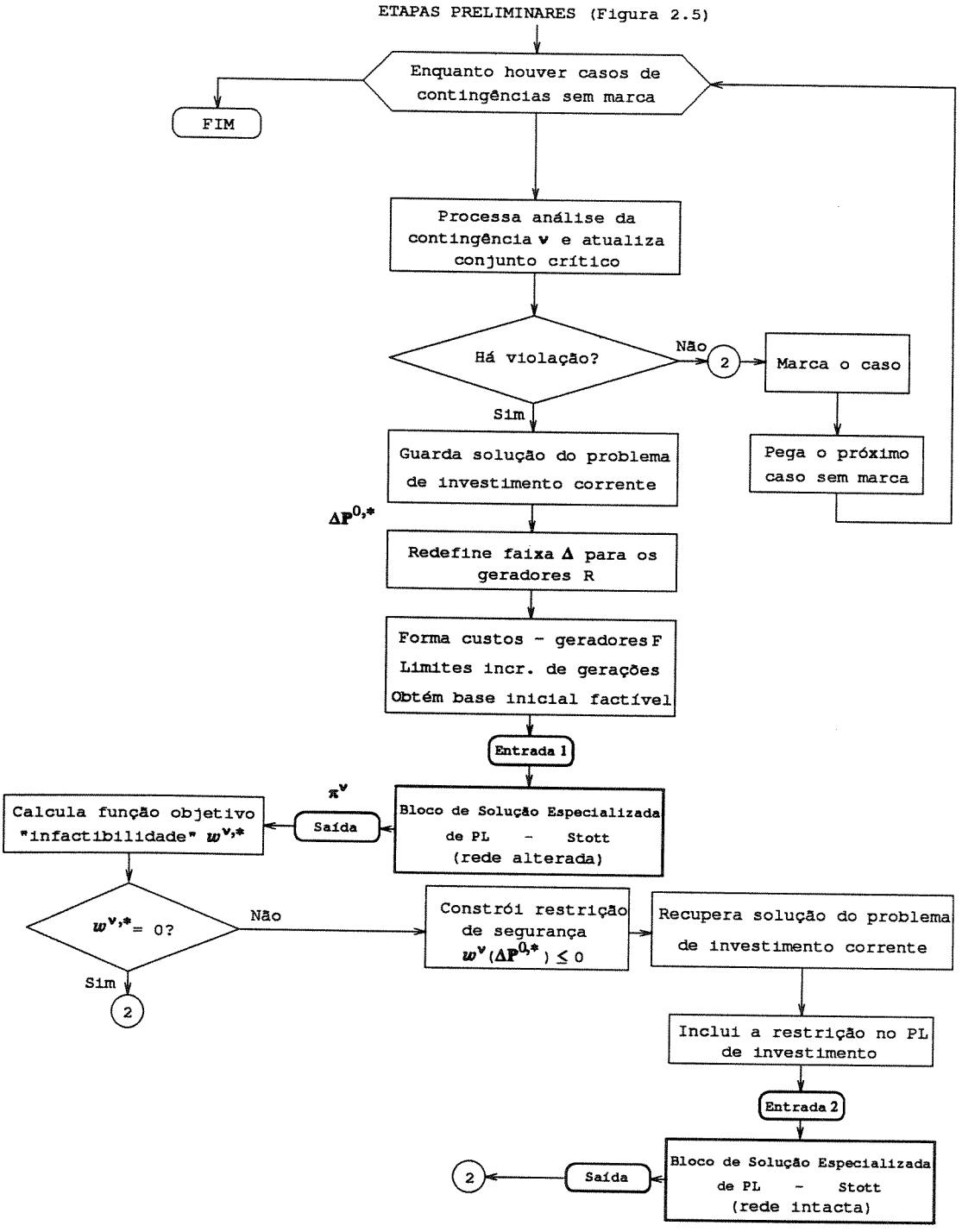


Figura 2.9: Diagrama de blocos do fluxo de potência ótimo com restrições de segurança com capacidade de remanejamento pós-contingência (FPORS-III).

Antes de passar à obtenção dos cortes alguns itens são estudados a seguir, como os geradores fictícios e seus usos.

Geradores Fictícios no Caso Geral

Pode-se dividir os geradores em dois conjuntos: reais e fictícios, respectivamente, simbolizados por R e F. Em todas as implementações de FPO elaboradas nesta tese são utilizados ostensivamente geradores fictícios em todas as barras da rede, seja barra típica de geração ou específica de carga. A Figura 2.10 ilustra os tipos de barras cobertos com geradores fictícios.

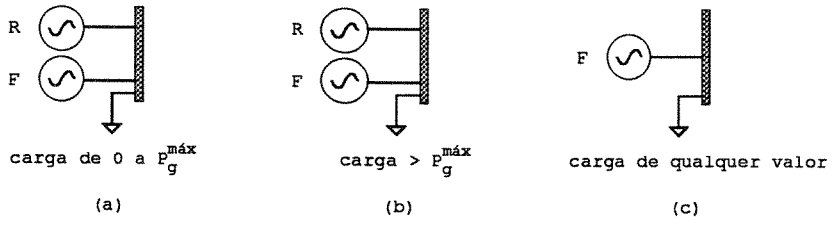


Figura 2.10: Geradores fictícios em todas as barras.

Os limites dos geradores ilustrados na Figura 2.10 são definidos do seguinte modo:

$$\text{Limites} \begin{cases} -9999 \leq P_g^F \leq 0 \quad [MW], & \text{para carga de 0 a } P_g^{\text{máx}} \\ 0 \leq P_g^F \leq +9999 \quad [MW], & \text{para carga} \quad > P_g^{\text{máx}} \\ 0 \leq P_g^F \leq +9999 \quad [MW], & \text{para barras de carga} \end{cases}$$

Para o correto funcionamento do algoritmo especializado de solução de PL é essencial assegurar que o produto custo versus P_g^F seja sempre não-negativo [9,12,35]. Neste aspecto, chama-se a atenção do leitor para o Apêndice D, onde estão mostradas as informações contidas num pequeno banco de dados. (Na verdade, só é necessário ler os dados dos geradores reais e definir como os geradores fictícios serão enquadrados no programa para cada sistema-teste.)

A introdução de geradores fictícios traz inúmeras vantagens para a implementação, dentre elas pode-se citar aumento da confiabilidade e garantia de robustez do programa, que são aspectos essenciais em qualquer processo de otimização, especialmente quando se pretende torná-lo operacional para sistemas reais de grande porte. Particularmente no caso da decomposição de Benders, as gerações fictícias são indispensáveis porque possibilitam a elaboração do programa.

Sendo j o índice do gerador, tem-se:

$$\Delta\mathbf{P} = \begin{array}{|c|c|c|c|c|}\hline \Delta P_1 & \Delta P_2 & \cdots & \Delta P_j & \cdots \\ \hline \end{array}\ t \qquad (j)$$

$$\mathbf{c} = \begin{array}{|c|c|c|c|c|}\hline c_1 & c_2 & \cdots & c_j & \cdots \\ \hline \end{array} \qquad (j)$$

Se j for um gerador tipo R ele será um gerador real e seu custo incremental é dado no vetor **c**. Caso contrário, j será tipo F, fictício, de custo incremental elevado (dado em **c**, na posição correspondente). O quanto c_j deve ser alto relativamente aos valores reais depende do problema em análise. Neste trabalho, na definição dos custos dos geradores fictícios, toma-se o cuidado de garantir que esses valores sejam superiores a qualquer um dos custos dos geradores reais. A meta é fazer com que o processo de otimização somente lance mão de geradores fictícios quando todos os recursos reais tiverem sido empregados dentro dos limites possíveis.

Convém ressaltar que, os geradores R são os equipamentos físicos realmente existentes no sistema ou, se não existem de fato, representam controles como, por exemplo, os defasadores. Já os geradores F (fictícios) são concebidos como um artifício matemático para permitir a continuidade dos cálculos mesmo quando o problema na realidade não possuir solução.

Função Objetivo do Subproblema de Operação - "Infactibilidade"

A função objetivo do subproblema de operação (2.15) é tal que os geradores reais têm custos nulos, e os geradores fictícios têm custos $+1$ ou -1 dependendo da faixa de limites do gerador considerado.

De (2.15), tem-se a função objetivo:

$$\mathbf{d}\Delta\mathbf{P}^{\nu} = \mathbf{d} \begin{array}{|c|}\hline \Delta P_1^{\nu} \\ \hline \vdots \\ \hline \Delta P_j^{\nu} \\ \hline \vdots \\ \hline \Delta P_n^{\nu} \\ \hline \end{array} = \sum_{\substack{\text{se } j\in R \\ \text{ou } j\in F}}^{n} d_j\, \Delta P_j^{\nu}$$

Onde,

$$d_j = \begin{cases} -1, & \text{se o gerador } j \text{ tipo } F \text{ estiver conectado como na Figura 2.10(a)} \\ 0, & \text{se o gerador } j \text{ for do tipo R} \\ +1, & \text{se o gerador } j \text{ tipo } F \text{ estiver conectado como nas Figuras 2.10(b)-(c)} \end{cases}$$

As variáveis de controle fictícias têm o papel de assegurar a factibilidade matemática do subproblema para qualquer ponto $\mathbf{\Delta P}^{0,*}$ dado, e elas podem ser interpretadas como a quantidade de violação associada ao ponto de operação pós-contingência $\mathbf{\Delta P}^{\nu}$. Se um problema for factível, estas variáveis serão naturalmente iguais a zero, o que implicará no valor objetivo do subproblema também nulo.

Para o subproblema de operação ν (2.15), pode-se concluir que:

$$w^{\nu} = 0 \iff \text{subproblema é factível}$$

$$w^{\nu} > 0 \iff \text{subproblema é infactível}$$

Obviamente, na prática utiliza-se uma tolerância (e.g., 10^{-3} ou outro valor adequado ao sistema sob teste) para que um subproblema seja rotulado de factível.

Construção dos Cortes de Benders

Após cada solução do problema de investimento tem-se $\mathbf{\Delta P}^{0,*}$. Resolvendo-se, em seguida, o subproblema de operação ν a partir de $\mathbf{\Delta P}^{0,*}$ obtém-se o valor da função objetivo $w^{\nu,*}$ e, também, os multiplicadores simplex da solução, π^{ν}. Este procedimento é feito para $\nu = 1, 2, \cdots, N_c$. Em particular, como está sendo usado o algoritmo especializado de solução de PL de Stott, os multiplicadores são os próprios elementos do vetor $\lambda = (\lambda^b \mid \lambda^g)$ naturalmente obtidos ao final da solução do PL de operação.

Assim, conhecidos os valores $w^{\nu,*}$, $\mathbf{\Delta P}^{0,*}$ e π^{ν}, uma aproximação linear para a função $w(\mathbf{\Delta P}^{0,*})$ é construída após a solução de cada subproblema de operação que não tenha passado no teste de factibilidade. Pode-se provar (vide Apêndice B) que os coeficientes desta aproximação linear são os próprios multiplicadores simplex (conhecidos também como variáveis duais, custos marginais ou multiplicadores de LaGrange) associados às restrições do subproblema de operação. Para o PL definido em (2.15), a forma específica do corte de Benders é dada por (2.17).

$$w^{\nu,*} + \pi^{\nu}\left(\mathbf{\Delta P}^{0} - \mathbf{\Delta P}^{0,*}\right) \leq 0 \qquad (2.17)$$

Onde,

$$\pi^{\nu} = \mathbf{d}[B]^{-1} \rightarrow \lambda = \left(\lambda^{b} \mid \lambda^{g}\right)$$

A expressão (2.17) é reescrita de maneira conveniente ao algoritmo de solução de PL aqui utilizado conforme indica (2.18):

$$\begin{aligned} \pi^{\nu}\mathbf{\Delta P}^{0} &\leq \pi^{\nu}\mathbf{\Delta P}^{0,*} - w^{\nu,*} \\ \mathbf{A}^{t}_{km}\mathbf{\Delta P}^{0} &\leq L^{\text{máx}} \end{aligned} \tag{2.18}$$

Onde,

$$\mathbf{A}^{t}_{km} = \pi^{\nu}$$

$$L^{\text{máx}} = \pi^{\nu}\mathbf{\Delta P}^{0,*} - w^{\nu,*}$$

A função $w^{\nu}(\mathbf{\Delta P}^{0,*}) \leq 0$ fornece informação acerca das conseqüências da decisão de operação $\mathbf{\Delta P}^{0,*}$ se a ν-ésima contingência ocorrer. A expressão (2.17) representa a mudança no corte de carga pós-distúrbio como uma função linear das mudanças no ponto de operação do caso-base. Portanto, os coeficientes desta restrição podem ser interpretados como "direções gradientes" para incrementar a segurança do sistema. Para tornar mais clara a exposição da técnica de Benders empregada no programa FPORS-III, o Apêndice D mostra um exemplo numérico para um sistema de 2 barras.

Comentários

Cabe aqui fazer uma observação comparando-se a restrição de Benders e a restrição de segurança pela máxima violação do método de Stott. A restrição de Benders traz informação sobre a rede como um todo, ou seja, cobre toda sua extensão geográfica. Já o corte pela máxima violação contém informação sobre os arredores do ramo onde ocorreu o distúrbio, isto é, possui natureza localizada.

Para gerar os cortes, com $\mathbf{\Delta}$ genérico, a técnica de Benders utiliza os multiplicadores duais das restrições dados pela solução do PL pós-contingência. Se a função objetivo do subproblema de operação for corte de carga (ou geração total fictícia) os multiplicadores (π) darão a sensibilidade entre o corte de carga e a geração:

$$\pi^{\nu}_{j} = \frac{\partial w^{\nu,*}}{\partial \Delta P_{j}}$$

Suponha que, em vez de minimizar o corte de carga (FPORS-III), o problema fosse minimizar a violação da linha com maior sobrecarga no estado pós-contingência. Ou seja, o objetivo seria anular a sobrecarga, mesmo que para isso fosse necessário usar gerações fictícias. Esta função objetivo é mais simples que o corte de carga geral. No caso do subproblema ativo ($P\theta$) poderia dar bons resultados, uma vez que, quando fosse resolvido o problema da linha mais violada, provavelmente as outras linhas teriam suas sobrecargas resolvidas ou praticamente resolvidas. Esta é a situação similar à abordagem dada no programa FPORS-II (ou no FPORS-I), que constitui a idéia de Stott baseada em relaxação.

No caso da função objetivo do PL pós-contingência ser a violação máxima, os multiplicadores serão dados pela derivada da violação máxima ($\Delta f_{km} = \mathbf{A}_{km}^t \mathbf{\Delta P}$) em relação às injeções de potência ativa ($\mathbf{\Delta P}$). O resultado é que o vetor multiplicador, neste caso, será o próprio $\mathbf{A}_{km}^t$.

Um aspecto interessante que está mostrado neste trabalho é que, para ambos os métodos (Benders ou Stott), o problema (2.9) e as soluções apresentadas no caso $\mathbf{\Delta} = \mathbf{0}$ recaem no problema (2.8), significando que os métodos Stott e Benders se tornam equivalentes para esta situação particular.

Deste resultado é possível inferir que o método de solução de Benders pode ser visto como uma técnica que unifica as soluções do problema de fluxo de potência ótimo com restrições de segurança, cobrindo os valores de $\mathbf{\Delta}$ desde zero até infinito.

2.4.5 Análise de Contingências

Ao trabalhar nas implementações de fluxo de potência ótimo com restrições de segurança é que se percebe a importância da análise de contingências no contexto das funções de um centro de controle. Não se sabe realmente o que é mais importante no binômio contingência-otimização.

Nesta tese, diversos algoritmos de análise de contingências foram implementados, testados e posteriormente paralelizados. Em seguida a esta fase é que foram elaboradas e introduzidas as rotinas de otimização.

Em algoritmos de análise de contingências é crucial saber resolver eficientemente sistemas esparsos de equações algébricas lineares. As modernas técnicas de exploração da esparsidade de matrizes e vetores foram utilizadas, bem como os métodos de compensação orientados à esparsidade e a técnica de atualização de fatores triangulares das matrizes de redes (vide Capítulo 3). De outra vertente, já na classe dos métodos iterativos de solução de $[A]\mathbf{x} = \mathbf{b}$, foram também pesquisados e implementados algoritmos alternativos. Tais métodos mostraram-se competitivos com os consagrados métodos diretos, embora ainda estejam em fase de pesquisa e melhoramento (vide Apêndice E).

Tendo em vista a importância da análise de contingências para este trabalho, o Capítulo 3

é inteiramente dedicado a este tópico. Como uma extensão da pesquisa neste assunto foram produzidas implementações concorrentes num ambiente de programação distribuída, as quais estão descritas no Capítulo 4 e, também, no Capítulo 5 juntamente com o FPORS.

Quanto aos métodos iterativos na solução de $[A]\mathbf{x} = \mathbf{b}$, o Apêndice E apresenta parte do trabalho desenvolvido, em especial a aplicação ao problema de *screening* de contingências.

2.4.6 Características Principais dos Programas Desenvolvidos

Descrição das Implementações Seqüenciais de FPORS

O cálculo do FPORS pode ser dividido nas seguintes etapas:

1. Leitura de dados da rede, dados de geradores e dados da lista de contingências;
2. Obtenção do ponto inicial de operação, que pode ser feita através de um fluxo de potência *AC* ou de um fluxo *DC*. Esta etapa é referida como inicialização *AC* ou inicialização *DC*, dependendo de como é feito este cálculo;
3. Cálculo do PL do caso-base (ou FPO caso-base), onde se busca a obtenção de uma solução considerando-se a rede intacta (i.e., sem alterações topológicas). Neste cálculo, todas as restrições de contingências são relaxadas, restando ao processo de otimização minimizar a função objetivo escolhida respeitando apenas os limites das variáveis de controle (geradores controláveis) e os limites de fluxos ativos em ramos;
4. Análise de todas as contingências especificadas na lista para o ponto de operação obtido da etapa precedente. Os estados pós-contingências são calculados através de compensação ou atualização de fatores triangulares da matriz $[B']$ e, a partir destes, são obtidos os índices de severidade de sobrecargas (índice este designado por PI^{ν}_{MW}, para o caso ν de contingência);
5. Ordenação dos casos da lista de contingências em ordem decrescente de severidade, de modo que, ao final desta etapa, os casos mais severos figurarão no topo da lista (tais casos são aqui denominados casos críticos); o critério para "filtrar" os casos críticos consiste em selecionar as contingências cujo índice seja superior a uma determinada tolerância (i.e., $PI^{\nu}_{MW} \geq \varepsilon$);
6. De posse da lista ordenada de contingências, onde os casos críticos situam-se no topo desta lista, inicia-se o processo iterativo do fluxo de potência ótimo com restrições de segurança; o processo iterativo do FPORS compreende a execução de cada caso crítico de contingência, verificando-se a sobrecarga provocada no sistema e formando-se um conjunto de ramos violados ou na iminência de violar (é o chamado conjunto crítico); restrições são construídas e impostas ao processo PL; à medida que novos pontos de operação são obtidos,

avançando-se em direção ao ponto ótimo-seguro, contingências devem ser analisadas e testadas quanto à violações de limites; o processo pára quando todas as contingências tiverem sido processadas em relação ao estado mais recente e nenhuma delas violar os limites da segurança do sistema;

6.1 Dado um caso, o teste de verificação da violação corresponde à análise da contingência em relação ao estado básico vigente;

6.2 A atualização do conjunto crítico será efetuada sempre que for detectada alguma sobrecarga (ou a iminência de sobrecarga) em qualquer ramo da rede elétrica;

6.3 Dentre os elementos do conjunto crítico, correspondente ao ramo, cuja violação é a máxima identificada, constrói-se uma restrição de segurança, que é função da gravidade da sobrecarga, dos parâmetros associados à topologia da rede e da sua localização;

6.4 De posse da restrição (ou corte) recém construída, tal restrição é introduzida no PL do caso-base (se FPORS-I ou FPORS-III) ou no PL de operação (se FPORS-II);

6.5 Ao incluir uma nova restrição no PL, o ponto de operação muda e conseqüentemente tem-se um novo estado operacional, a partir do qual as contingências da lista devem ser reavaliadas;

7. Processadas as contingências críticas da lista, restam os casos cuja avaliação prévia feita nas etapas 4 e 5 não indicou naquela ocasião sobrecargas; é importante esta verificação tendo em vista que o estado então disponível (obtido na etapa 6) é em geral diferente daquele estado da etapa 3; contingências que antes não violavam podem, agora, levar a sérias violações da segurança. Diante deste fato, procede-se à análise das contingências da lista excluindo-se obviamente aquelas já submetidas ao processo iterativo do FPORS; na eventualidade de alguma contingência indicar violação (ou de várias contingências) altera-se o *ranking* atual introduzindo-a logo após à última contingência crítica já considerada (isto deve ser feito sem perder informações da classificação atual, e todo este processo de análise de contingências deve ser realizado de forma rápida e confiável);

8. No caso da classificação ter sido atualizada em decorrência do processamento da etapa anterior, retorna-se à etapa 6 com o novo grupo de casos críticos; em caso contrário, ou seja, se novas violações não foram detectadas na etapa 7, o problema está terminado e o ponto de operação obtido é ótimo-seguro (no sentido de que os cenários pós-contingências são viáveis, assim como também a rede intacta, e a solução miminiza o objetivo previamente escolhido).

Controle de Casos Processados da Lista e Critério de Parada: Técnica de Marcas

Todos os programas de FPORS operam segundo uma lógica comum: para cada novo ponto obtido no bloco de otimização (i.e., novo caso-base) deve-se processar análise de contingência. Durante este processo intensivo de cálculos é essencial garantir que a análise de contingência seja sempre efetuada em relação ao caso-base mais recente. Mediante esta constatação, o ponto

ótimo-seguro será alcançado quando todos os cenários de contingência tiverem sido processados em relação ao mesmo caso-base, sem violações nos estados pós-contingências.

Isto posto, uma técnica de marcas, desenvolvida em [56], foi testada e adotada neste trabalho. Cada contingência (por exemplo, o caso ν) será identificada por um par-ordenado (ν, i_θ), que associa o estado básico (de índice i_θ) e a contingência ν. Assim, i_θ será o índice do estado (i.e., um contador) para o qual essa contingência foi analisada.

Para controlar os casos que já tenham sido processados (e em relação a qual estado eles foram calculados), forma-se um vetor de marcas, que é previamente inicializado com -1. Toda vez que se obtém um novo estado incrementa-se de um o valor do índice i_θ; isto ocorre quando se detecta violação e uma restrição de contingência é adicionada à base do processo PL. Sempre que uma contingência (ν) é testada e ela não leva à violação, a correspondente posição ν do vetor de marcas é imediatamente preenchida com o índice do estado em relação o qual a contingência foi calculada; o conteúdo do vetor de marcas da posição ν passa a ser i_θ.

Uma contingência só é processada se não coincidir o conteúdo da respectiva posição no vetor de marcas com o índice do estado atual.

Este procedimento de marcas de casos de contingências já analisados tem lugar na etapa 6 da descrição da implementação de FPORS feita anteriormente.

Geradores Fictícios

Conforme foi explicado anteriormente, geradores fictícios foram utilizados nos programas elaborados nesta tese. Esses elementos artificiais foram incorporados naturalmente ao algoritmo de solução de PL de Stott. Para isso, foi necessária a criação de dois novos apontadores e de um vetor de tipos (R ou F) e alguns cuidados adicionais na geração de restrições e no cálculo do estado incremental $\Delta\theta$. De acordo com a Figura 2.10, os geradores fictícios tiveram seus valores limites definidos de modo a assegurar a autonomia total da barra. Isto significa que, na situação extrema (hipotética) de supressão de todas as capacidades de transmissão da rede (por exemplo, extração dos ramos), as cargas ainda assim seriam atendidas, ou seja, o processo PL através de um despacho de "ordem de mérito" garantiria o balanço de potência, lançando mão dos onerosos recursos fictícios.

Esparsidade de Vetores e de Matrizes

Em todas os programas de análise de contingências e fluxo de potência ótimo foram empregadas técnicas de exploração de esparsidade de matrizes de redes. Associados à esparsidade de matrizes foram também utilizadas as técnicas de vetores esparsos (em particular, nas operações de substituição direta – *fast-forward*) e, como conseqüência da obtenção da árvore de fatoração de matrizes, o procedimento de atualização de fatores triangulares foi implementado no cálculo de contingências (i.e., alterações em ramos) [1,4,5,21,22,24,65].

Contingências Simples e Múltiplas de Ramos

A análise de contingências, seja na fase de seleção ou de avaliação e na construção de restrições (nas implementações segundo a metodologia de Stott), compreende os estudos relativos à saída de linha de transmissão ou de transformador. A saída simultânea de dois ou mais ramos é referida, neste trabalho, como contingência múltipla e a saída de um único ramo é a contingência simples. Contingências de geradores ou de cargas não foram estudadas.

Abordagem Seletiva da Análise de Contingências

Diversas técnicas de análise de contingências estão disponíveis na literatura. Na classe dos métodos diretos, onde se objetiva determinar por algoritmos de soluções aproximadas ou não o estado pós-contingência, três técnicas foram implementadas: (1) pós-compensação para contingências simples; (2) *mid*-compensação para contingências que correspondam à saída simultânea de dois, três ou quatro ramos; e (3) atualização de fatores triangulares, que reutiliza a estrutura criada para se trabalhar com vetores esparsos, para contingências de ordens superiores a quatro [21,24].

Funções Objetivos Disponíveis

O usuário dos programas pode optar por uma das seguintes funções objetivos.

Custo de operação (opção E):

$$\text{Função objetivo } f(\mathbf{P}^0) = \sum_j a_{1j} + a_{2j}P_j + a_{3j}P_j^2, \qquad \text{para } j \text{ tipo } R \tag{2.19}$$

Onde, $P_j = P_j^{\text{inicial}} + \Delta P_j$. Os coeficientes a_{1j}, a_{2j} e a_{3j} são lidos do banco de dados do sistema sob análise, para cada gerador j. A parcela da função objetivo correspondente aos geradores fictícios (i.e., tipo F) é definida da seguinte forma: $\sum_j$(coeficiente arbitrário e alto) $\times P_j$.

Desvio quadrático (opção MDQ):

$$\text{Função objetivo } f(\mathbf{P}^0) = \sum_j \alpha_j \Delta P_j^2, \qquad \text{para todo } j \tag{2.20}$$

Na expressão (2.20), o coeficiente α_j é definido para os geradores reais do seguinte modo:

$$\alpha_j \triangleq \frac{1}{(P_j^{\text{máx}})^2}$$

Sendo $P_j^{\text{máx}}$ o limite superior do gerador j.

Definir α_j desta maneira é uma escolha interessante, porque tende a penalizar os geradores de menor capacidade de geração em detrimento das máquinas de maior porte. Ou seja, o mecanismo de otimização utiliza, em primeiro lugar, os recursos das máquinas de maior capacidade.

Os geradores fictícios têm os coeficientes α_j definidos sempre superiores aos valores dos coeficientes dos geradores reais; por exemplo, como o supremo dentre os coeficientes de custos dos geradores reais.

Ressalta-se na expressão (2.20) que ΔP_j, onde $\Delta P_j = P_j - P_j^{\text{inicial}}$, denota o afastamento de P_j em relação ao ponto inicial. O ponto inicial pode ser obtido de um fluxo de potência, como pode ser um ponto especificado.

2.4.7 Um Exemplo de Como Opera o Processo Iterativo do FPORS no Modo Seqüencial

Suponha um banco de dados com uma lista de 100 contingências. Lidos os dados e calculado o ponto inicial passa-se à execução do FPO caso-base. Ao solucionar este FPO, onde são relaxadas todas as restrições de contingências, incrementa-se o contador i_θ (agora, i_θ vale 1).

Em relação ao estado obtido da solução do FPO caso-base são processadas todas as contingências da lista para se formar uma classificação das mesmas em ordem decrescente de severidade de sobrecargas. Tomando-se uma tolerância, por exemplo, $\varepsilon = 10^{-3}$, aquelas contingências cujos índices de severidade de sobrecargas excedem este patamar são consideradas críticas e serão utilizadas na produção de restrições de segurança no processo iterativo do FPORS. A Figura 2.11 ilustra a situação dos vetores de classificação e de marcas das contingências neste estágio de solução do problema. Conforme ilustra esta figura, as contingências cujas ordens são $\nu = 1, 2, \cdots, 7$ são processadas no FPORS; são as chamadas contingências críticas.

O processamento se faz do seguinte modo:

Primeira vez:

a. Toma-se $\nu = 1$; sua marca não coincide com o valor atual de i_θ (a marca é -1 e i_θ vale 1), portanto, calcula-se a contingência e constata-se violação; em seguida, atualiza-se o conjunto crítico;

b. Toma-se o ramo do conjunto crítico cuja violação é máxima e constrói-se uma restrição de segurança;

c. De posse da restrição, resolve-se o PL com a nova restrição incluída no processo de otimização; em seguida, incrementa-se o contador i_θ;

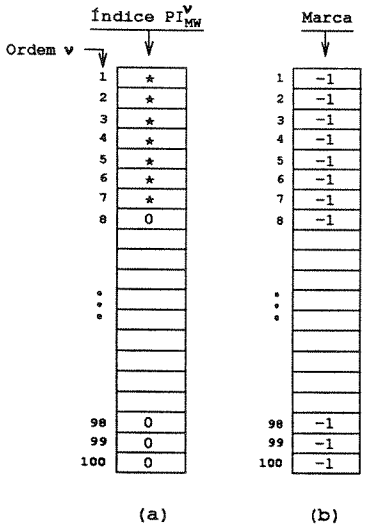


Figura 2.11: Situação dos vetores de *ranking* e de marcas antes do processo iterativo do FPORS: (a) lista ordenada das contingências; (b) vetor de marcas.

d. Marca-se a contingência ν calculada, assim:

$$marca(\nu) \leftarrow i_\theta$$

e. A contingência recém processada nunca será logo em seguida reutilizada para produção de restrição uma vez que sua marca coincidirá com o valor atual do contador i_θ; então toma-se a contingência seguinte (ou seja, desloca-se na lista de casos críticos para a posição sucessiva);

f. Procede-se para $\nu = 2$ de modo análogo ao caso $\nu = 1$ a partir do passo **a.** É oportuno ressaltar que, se nova restrição for construída em função da contingência $\nu = 2$, o cálculo feito para a contingência precedente já estará desatualizado, contudo, a técnica de marcas exigirá que $\nu = 1$ seja posteriormente processada em relação ao caso-base vigente.

Os passos descritos acima são executados para todas as contingências críticas, no presente exemplo, de $\nu = 1$ a $\nu = 7$. Após serem percorridos os passos acima, as marcas das contingências estarão como ilustra a Figura 2.12(a). Verifica-se que as marcas estão diferentes, indicando que as contingências foram processadas relativamente a casos-bases distintos, portanto, o critério de parada não foi ainda satisfeito.

Tendo em vista o não atendimento do critério de parada (uma vez que as marcas são distintas), os casos críticos de contingências devem ser novamente processados para verificação de violação e eventual construção de restrição de segurança. Isto posto, repete-se o procedimento

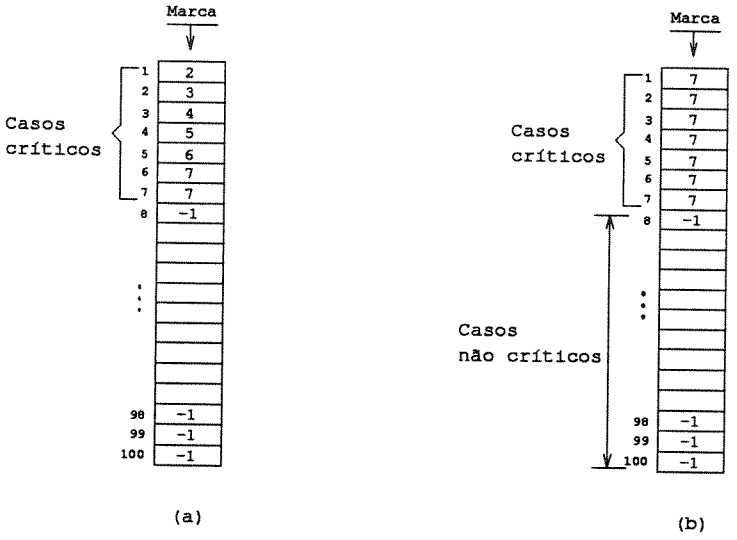


Figura 2.12: Situação do vetor de marcas durante o processo iterativo do FPORS: (a) vetor de marcas após a execução dos passos a. a f. pela primeira vez; (b) vetor de marcas após a execução dos passos a. a f. pela segunda vez.

iterativo do FPORS para as contingências $\nu = 1, 2, \cdots, 7$:

Segunda vez:

a. Toma-se $\nu = 1$; sua marca não coincide com o valor atual de i_θ (a marca é 2 e i_θ vale 7), portanto, calcula-se a contingência e constata-se violação; em seguida, atualiza-se o conjunto crítico; os passos que se seguem, **b.**, **c.**, **d.**, **e.** e **f.** são análogos aos passos descritos acima.

Ao finalizar o processamento pela segunda vez, para este exemplo em particular, as marcas das contingências críticas ficam conforme ilustra a Figura 2.12(b). Observa-se desta figura que as marcas são as mesmas para os casos críticos, embora ainda sejam iguais a -1 para as contingências que não foram classificadas como críticas. Isto requer que as contingências a partir da 8ª posição do *ranking* original sejam verificadas quanto à violação relativamente ao ponto de operação atual (cujo contador é $i_\theta = 7$).

Análise das contingências não críticas (vide Figura 2.12(b)):

As contingências cujas ordens são $\nu = 8, 2, \cdots, 100$ devem ser analisadas em relação ao último estado obtido do processo iterativo do FPORS. Uma dentre duas ocorrências é possível nesta etapa: (1) nenhuma das contingências não críticas apresenta violação pós-contingência; (2) pelo menos uma contingência (por exemplo, a contingência $\nu = 98$) apresenta violação.

Ocorrência 1 - nenhuma contingência apresenta violação:

Esta situação é a mais favorável porque significa que o estado operacional obtido no processo iterativo do FPORS atende também às demais contingências da lista. O problema está pronto e a solução é aquela indicada com o mais recente valor de i_θ. A Figura 2.13 mostra como fica o vetor de marcas ao final da solução.

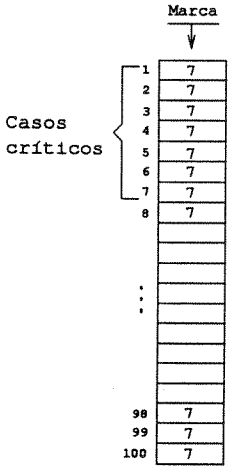


Figura 2.13: Situação do vetor de marcas ao final do processo iterativo do FPORS quando nenhuma contingência apresenta violação (ocorrência 1).

Ocorrência 2 - pelo menos uma contingência apresenta violação:

A Figura 2.14 mostra esta ocorrência, onde uma contingência antes não crítica leva à violação em relação ao estado atual pós-processo iterativo do FPORS. Esta situação requer o re-processamento do FPORS com o novo conjunto de casos críticos.

A atualização do conjunto de casos críticos é feita trocando-se a posição da contingência que apresentou violação pela posição da contingência não crítica situada logo após os casos críticos originais. A Figura 2.15 ilustra a atualização dos casos críticos; no exemplo, a contingência $\nu = 98$ passa a integrar a lista de casos críticos na 8ª posição; a contingência $\nu = 8$ assume a 98ª posição no *ranking*.

Agora, as contingências cujas ordens são $\nu = 1, 2, \cdots, 8$ são processadas no FPORS; são as chamadas contingências críticas. Repete-se o processo descrito acima até que todas as contingências da lista (isto é, os 100 casos) estejam marcados com o mesmo valor de i_θ.

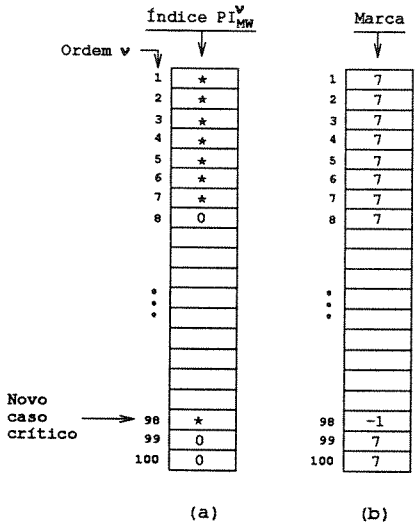


Figura 2.14: Situação dos vetores de *ranking* e de marcas antes de finalizar o processo iterativo do FPORS (ocorrência 2): (a) lista ordenada das contingências; (b) vetor de marcas.

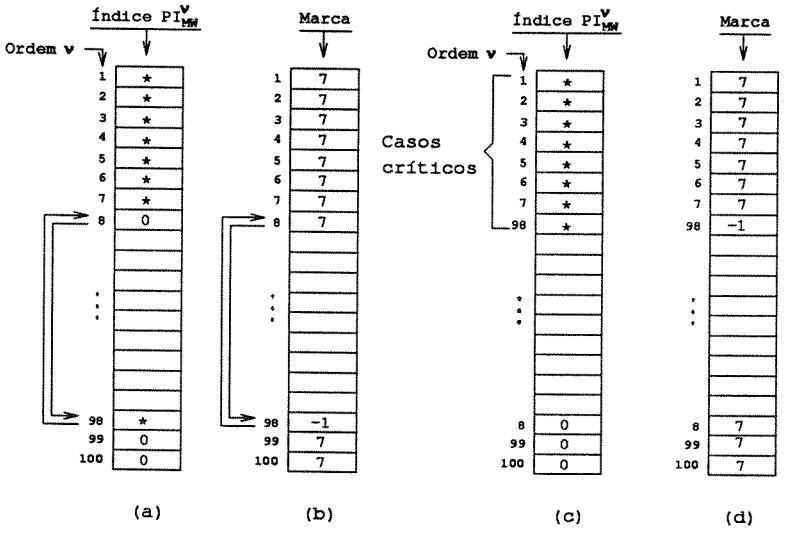


Figura 2.15: Mecanismo de atualização dos vetores de *ranking* e de marcas quando da ocorrência 2: (a) lista ordenada das contingências; (b) vetor de marcas; (c) e (d) mostram a lista ordenada das contingências e o vetor de marcas após a atualização.

Capítulo 3

ANÁLISE DE CONTINGÊNCIAS

3.1 Introdução

Análise de contingências são estudos que envolvem a simulação da ocorrência de distúrbios na rede elétrica. Esses estudos são realizados para conhecer previamente o comportamento do sistema em situações que impliquem em saídas/entradas de componentes e, a partir dessas informações, adotar medidas que garantam a continuidade e a qualidade do fornecimento.

Para grandes sistemas elétricos, a exigência da análise completa de todas as prováveis saídas de equipamentos implicaria no processamento de várias centenas de casos de contingências a freqüentes intervalos de tempo. O processamento de cada situação de contingência envolve a modificação da rede do caso-base para simular a saída do equipamento, para isso se faz necessária a solução de um fluxo de potência seguido da verificação de violações nos estados pós-contingências. Se todos os casos de contingências tivessem que ser processados num tempo muito curto, a carga computacional seria altíssima. Além disso, uma exaustiva avaliação de contingências (i.e., cálculos pós-seleção de contingências) é muito ineficiente, uma vez que, na prática, a maioria das contingências não representa qualquer ameaça à segurança do sistema. Portanto, um significativo ganho pode ser conseguido simplesmente estudando-se apenas os casos realmente importantes.

Um grande número de técnicas de análise de contingências vêm sendo pesquisadas e algumas delas já são utilizadas nos Centros de Controle das empresas de energia elétrica, porém, até o presente momento, nenhuma mostra absoluta vantagem sobre as demais. Consistentes esforços têm sido empregados para aumentar a velocidade de processamento das tarefas de seleção e avaliação de contingências. Significativos desenvolvimentos na área de seleção de contingências incluem ordenação de contingências através de índices de *performance*, soluções locais baseadas em relaxação concêntrica, e métodos de vizinhança em geral. Por outro lado, importantes

desenvolvimentos em análise de contingências em geral incluem a introdução do fluxo de carga desacoplado rápido, esparsidade orientada aos métodos de compensação, métodos de vetores esparsos, e refatoração parcial [21,22,24,42,54].

De outra parte, a partir de meados da década de '80, surgiram soluções do problema de análise de contingências que aliam a eficiência das técnicas acima mencionadas ao uso intensivo dos emergentes recursos computacionais de *hardware* de alto desempenho [40,41,55,57,60,61].

Colocada a importância da seleção de contingências, um módulo de segurança estática de um moderno Centro de Controle deve possuir *softwares* capazes de detectar em tempos muito curtos os casos de contingências mais severos para um ponto de operação especificado. Podem ser destacados dois requisitos de um módulo de análise de segurança:

- A partir do instante de uma mudança no sistema de potência (instante caracterizado pelo *breaker operation*), ao operador deve ser dado a conhecer dentro de, por exemplo, cinco minutos, quais são as piores contingências;
- Uma vez que as piores contingências são conhecidas, uma estratégia corretiva de cálculo dever ser efetuada (dentro de, por exemplo, outros cinco minutos), indicando a melhor combinação dos controles para sanar as piores contingências.

Numa fase preliminar, deseja-se saber quais casos levam a sobrecargas e quais suas gravidades para o sistema. Neste aspecto, são úteis, como medidas relativas das gravidades das contingências, os índices de severidade. De outro lado, o planejamento preventivo da operação procura responder ao que fazer para corrigir a violação caso a contingência postulada venha a ocorrer. O planejamento preventivo é essencialmente um problema de determinar estratégias de controle; sabendo-se que este problema quando formulado sem função objetivo é pobre, é sugerida sua abordagem através de solução de PL. Esta segunda fase é conhecida como determinação da estratégia corretiva das violações. Com base nestas idéias é que se propõe neste capítulo a construção de um *ranking* (que será obtido pós-verificação das sobrecargas) derivado da capacidade corretiva do sistema. O índice usado neste caso é obtido da própria função objetivo desvio quadrático do FPO calculado nas condições pós-contingência da rede.

Este capítulo compõe-se das seguintes partes. As técnicas de análise de contingências baseadas em compensação e refatoração parcial são apresentadas (seção 3.2). Toda a formulação é fundamentada nas equações do fluxo de potência desacoplado rápido [8,20] e em linearizações do modelo da rede. São estudados os métodos de pós-compensação, *mid*-compensação, atualização de fatores triangulares, *DC* incremental e, também, os métodos de *screening* $1P\theta$ e $1P\theta - 1QV$. São discutidos dois índices de severidade que são usados na formação de *ranking* de contingências. Na seção 3.3 são apresentadas as expressões das restrições de contingências do fluxo de potência ótimo. Por último, a seção 3.4 traz uma nova proposta de obtenção de um *ranking* das contingências, que seria obtido pós-seleção de contingências (portanto, após a detecção das violações mais severas). Os cálculos dos índices para formar esta segunda lista são efetuados por programação linear com a rede alterada, retratando a contingência sob análise. O valor ótimo da função objetivo deste PL é tomado como índice de *performance* (designado aqui como PI_{MDQ}), que é então usado para formar o segundo *ranking*.

3.2 Técnicas de Análise de Contingências.

A área de sistemas de energia elétrica, especialmente os estudos de redes, é um campo onde o uso de técnicas de esparsidade é indispensável. É comum, mesmo em problemas cuja formulação é não-linear, o aparecimento de sistemas algébricos lineares com matriz de coeficientes esparsa e com vetor independente de pouquíssimas posições preenchidas. Diante desse fato, a exploração da esparsidade é requerida sob pena de se inviabilizar a aplicação de um dado algoritmo. A aplicação de técnicas de esparsidade torna mais complexa a programação, entretanto, o benefício alcançado é substancialmente grande, e está fartamente documentada em [1,4,5,20,21,22,24,28,29,54,65].

Nos anos '60 surgiram as técnicas de esparsidade de matrizes, que, desde então, vêm sendo aperfeiçoadas e utilizadas. Somente nos anos '80 é que foram difundidos os métodos de vetores esparsos, que permitem acelerar consideravelmente a resolução de sistemas de equações lineares. Acompanhando de perto o *boom* dos métodos de vetores esparsos, foram divulgados métodos de refatoração parcial e/ou de atualização de fatores, que se "casam" perfeitamente com os conceitos associados à exploração da esparsidade de matrizes e vetores.

Esta seção tem por objetivo descrever a associação destas metodologias na resolução do problema de análise de contingências. Porém, antes de passar adiante são dadas algumas definições:

É comum aparecer nos estudos de análise de contingências o vetor $\mathbf{e}_{ij}$ e a matriz $[M]$. $\mathbf{e}_{ij}$ é definido da seguinte forma para a saída de um ramo cujas barras extremas sejam i e j.

$$\mathbf{e}_{ij} \triangleq \begin{array}{|c|l} \hline 0 & \\ \hline \vdots & \\ \hline +1 & i \\ \hline \vdots & \\ \hline -1 & j \\ \hline \vdots & \\ \hline 0 & n \\ \hline \end{array}$$

A matriz $[M]$ é esparsa (i.e., $[M']$ ou $[M'']$ se são referentes a $[B']$ ou a $[B'']$), $(n \times q)$, geralmente com $n >> q$, constituída por no máximo dois elementos não-nulos em cada coluna: $+1$ e -1; com n igual ao número de barras e q a ordem da contingência (isto é, a quantidade de ramos alterados simultaneamente). $[M]$ pode ser expressa em função dos vetores $\mathbf{e}_{i_l j_l}$, para $l = 1, 2, \cdots, q$:

$$[M] = \begin{array}{|c|c|c|c|c|c|} \hline \mathbf{e}_{i_1 j_1} & \mathbf{e}_{i_2 j_2} & \cdots & \mathbf{e}_{i_l j_l} & \cdots & \mathbf{e}_{i_q j_q} \\ \hline \end{array}$$

Partindo-se da premissa de que a solução do caso-base em relação a qual serão simuladas as contingências foi obtida, estarão disponíveis as tensões, em módulo e ângulo, V_k^0 e θ_k^0, de todas as barras para esta condição específica de operação. Estarão também disponíveis as matrizes $[B']$ e $[B'']$, supostas simétricas, na forma fatorada: $[B'] = [L'][D'][L']^t$ e $[B''] = [L''][D''][L'']^t$.

As equações incrementais das iterações $P\theta$ e QV do fluxo de potência desacoplado rápido são: $\mathbf{\Delta P} = [B']\mathbf{\Delta\theta}$ e $\mathbf{\Delta Q} = [B'']\mathbf{\Delta V}$. Aqui são omitidas as normalizações pelas magnitudes das tensões nodais apenas por simplicidade [20].

3.2.1 Métodos Baseados em Compensação

A compensação é utilizada em relação ao cálculo dos vetores de correções $\mathbf{\Delta\theta}$ e $\mathbf{\Delta V}$.

Métodos de compensação (sejam eles derivados da aplicação do Teorema da Compensação – através de argumentações físicas – ou a partir de deduções puramente algébricas), essencialmente, consistem na utilização do Lema de Inversão de Matrizes (LIM) [5,20,21]. Neste contexto, compensação resume-se em exprimir a inversa da matriz alterada em termos da inversa da matriz original. Cabe esclarecer que o termo inversa de uma matriz é apenas uma figura de linguagem, uma vez que, na verdade, as matrizes esparsas são decompostas em seus fatores triangulares inversos.

As matrizes pós-contingências podem ser expressas em função de $[B']$ e $[B'']$ do caso-base pela aplicação do Lema de Inversão de Matrizes. Isto permite escrever as equações incrementais do fluxo de potência desacoplado rápido nas formas dadas em (3.1) e (3.2).

$$\mathbf{\Delta\theta} = \left\{[B']^{-1} - [B']^{-1}[M'][C'][M']^t[B']^{-1}\right\}\mathbf{\Delta P} \tag{3.1}$$

$$\mathbf{\Delta V} = \left\{[B'']^{-1} - [B'']^{-1}[M''][C''][M'']^t[B'']^{-1}\right\}\mathbf{\Delta Q} \tag{3.2}$$

Onde,

$$[C'] = \left[[W'_B] + [M']^t[B']^{-1}[M']\right]^{-1}$$

$$[C''] = \left[[W''_B] + [M'']^t[B'']^{-1}[M'']\right]^{-1}$$

Neste equacionamento não é relevante a normalização por $\mathbf{V}$; contudo, na programação do processo iterativo é relevante, pois o efeito numérico dessa normalização repercute na velocidade de convergência.

Os termos $[M']^t\ [B']^{-1}\ [M']$ e $[M'']^t\ [B'']^{-1}\ [M'']$ possuem o significado de (matrizes de) impedâncias equivalentes vistas das barras terminais do ramo contingenciado. Já os termos $[W'_B]$ e $[W''_B]$ são, no caso geral de q alterações, matrizes diagonais $q \times q$, cujos elementos são, respectivamente, $\frac{1}{b_{i_l j_l}}$ e $-x_{i_l j_l}$, com $l = 1, 2, \cdots, q$ (onde, $x_{i_l j_l}$ é a impedância do ramo $i_l - j_l$ contingenciado, e $b_{i_l j_l}$ sua susceptância). Caso a versão do desacoplado rápido usada seja a XB, os elementos dessas matrizes serão respectivamente dados por $-x_{i_l j_l}$ e $\frac{1}{b_{i_l j_l}}$.

A partir das equações (3.1)-(3.2) são derivados os métodos de compensação orientados à exploração da esparsidade de matrizes e vetores, conforme expõe [21]. Esses métodos diferem quanto à seqüência em que são efetuadas as operações indicadas em (3.1)-(3.2) e, também, se distinguem quanto ao aspecto desempenho no tratamento de contingências simples e múltiplas.

Respeitando-se as especificidades de cada método, neste trabalho foram implementados os seguintes procedimentos: pós-compensação e *mid*-compensação. Sendo que, o primeiro foi utilizado na análise de contingências simples de ramo e, o segundo, na análise de contingências múltiplas de ramo de ordens dois, três e quatro. Para contingências de ordens superiores a quatro, aplicou-se a atualização de fatores (vide Figura 3.3 na seção 3.2.5). Os métodos de compensação são descritos a seguir; são salientadas duas fases de cálculo: (1) preparatória; e (2) de solução.

A) Pós-compensação para Contingência Simples

As expressões (3.1)-(3.2) são particularizadas para a situação de contingência simples (i.e., saída do ramo $i - j$):

$$\Delta\boldsymbol{\theta} = \left\{[B']^{-1} - [B']^{-1}\,\mathbf{e}'_{ij}\,c'\,(\mathbf{e}'_{ij})^t\,[B']^{-1}\right\}\,\Delta\mathbf{P} \tag{3.3}$$

$$\Delta\mathbf{V} = \left\{[B'']^{-1} - [B'']^{-1}\,\mathbf{e}''_{ij}\,c''\,(\mathbf{e}''_{ij})^t\,[B'']^{-1}\right\}\,\Delta\mathbf{Q} \tag{3.4}$$

Onde, c' e c'' são, neste caso, simples escalares:

$$c' = \left[w'_B + (\mathbf{e}'_{ij})^t\,[B']^{-1}\,\mathbf{e}'_{ij}\right]^{-1}$$

$$c'' = \left[w''_B + (\mathbf{e}''_{ij})^t\,[B'']^{-1}\,\mathbf{e}''_{ij}\right]^{-1}$$

Sejam os vetores $\Delta\boldsymbol{\theta}_a$ e $\Delta\mathbf{V}_a$ definidos conforme as expressões (3.5)-(3.6):

$$[B']\Delta\boldsymbol{\theta}_a \triangleq \Delta\mathbf{P} \tag{3.5}$$

$$[B'']\Delta\mathbf{V}_a \triangleq \Delta\mathbf{Q} \tag{3.6}$$

As expressões (3.3)-(3.4) podem ser reescritas na forma pós-compensada, como (3.7)-(3.8):

$$\Delta\theta = \Delta\theta_a - [B']^{-1}\, \mathbf{e}'_{ij}\, c'\, (\mathbf{e}'_{ij})^t \Delta\theta_a \tag{3.7}$$

$$\Delta\mathbf{V} = \Delta\mathbf{V}_a - [B'']^{-1}\, \mathbf{e}''_{ij}\, c''\, (\mathbf{e}''_{ij})^t \Delta\mathbf{V}_a \tag{3.8}$$

Agora, é interessante estabelecer como ficam as expressões para o cálculo das impedâncias equivalentes (w'_T e w''_T) na situação particular de contingência simples. Neste caso específico, w'_T e w''_T são simples escalares e para obtê-los basta avaliar as expressões (3.9) e (3.10):

$$w'_T = (\mathbf{e}'_{ij})^t\, [B']^{-1} \mathbf{e}'_{ij} \tag{3.9}$$

$$w''_T = (\mathbf{e}''_{ij})^t\, [B'']^{-1} \mathbf{e}''_{ij} \tag{3.10}$$

Todos os cálculos devem ser efetuados sem perder de vista a característica de esparsidade.

Algoritmo Pós-compensação (Análise Completa)

Fase preparatória:

1. Resolver os sistemas lineares:

$$[B']\mathbf{w}'_{ij} = \mathbf{e}'_{ij}$$

$$[B'']\mathbf{w}''_{ij} = \mathbf{e}''_{ij}$$

   Tendo em vista a esparsidade do vetor independente aplica-se na programação os métodos de vetores esparsos;

2. Calcular as impedâncias equivalentes a partir da solução dos sistemas lineares do passo anterior:

$$w'_T = (\mathbf{e}'_{ij})^t \mathbf{w}'_{ij}$$

$$w''_T = (\mathbf{e}''_{ij})^t \mathbf{w}''_{ij}$$

3. Calcular os escalares c' e c'':

$$c' = \frac{1}{w'_B + w'_T}$$

$$c'' = \frac{1}{w''_B + w''_T}$$

   Aqui, pode-se detectar as contingências que conduzem a ilhamentos do sistema, entretanto, isto deve ser feito antes da análise completa das contingências. Numa etapa preliminar à análise AC, efetua-se um pré-processamento da lista original ("bruta"), onde, além de outras operações, são separados os casos que conduzem a ilhamentos da rede.

Concluída a fase preparatória, entra-se no processo iterativo normal do fluxo de potência [20], onde, a cada reavaliação de ângulos ou magnitudes de tensões, efetuam-se os passos da fase de solução descritos a seguir.

Fase de solução:

1. Calcular os vetores de *mismatch*, $\Delta\mathbf{P}$ e $\Delta\mathbf{Q}$.
2. Resolver os sistemas algébricos lineares (3.5)-(3.6) para obter $\Delta\theta_a$ e $\Delta\mathbf{V}_a$.
3. Calcular as correções $\Delta\boldsymbol{\theta}$ e $\Delta\mathbf{V}$, através das equações:

$$\Delta\boldsymbol{\theta} = \Delta\boldsymbol{\theta}_a - \left[\mathbf{e}'_{ij}(i)\Delta\theta_a(i) + \mathbf{e}'_{ij}(j)\Delta\theta_a(j)\right] c'\mathbf{w}'_{ij}$$

$$\Delta\mathbf{V} = \Delta\mathbf{V}_a - \left[\mathbf{e}''_{ij}(i)\Delta\mathbf{V}_a(i) + \mathbf{e}''_{ij}(j)\Delta\mathbf{V}_a(j)\right] c'\mathbf{w}''_{ij}$$

   Onde, $\Delta\theta_a(i)$ corresponde ao conteúdo da posição i do vetor $\Delta\theta_a$, e assim por diante;
4. Atualizar, ao final de cada meia iteração (ativa e reativa), o respectivo vetor de estado:

$$\boldsymbol{\theta} = \boldsymbol{\theta}^0 + \Delta\boldsymbol{\theta}$$

$$\mathbf{V} = \mathbf{V}^0 + \Delta\mathbf{V}$$

As iterações ativa ($P\theta$) e reativa (QV) devem ser efetuadas de forma alternada, conforme está estabelecido em [20], de acordo com o algoritmo do fluxo de potência desacoplado rápido.

B) *Mid*-compensação para Contingência Múltipla

Tomando-se a expressão (3.1), e supondo que a matriz $[B']$ tenha sido fatorada na forma $[B'] = [L'][D'][U']$, é possível transcrevê-la como a seguir:

$$\Delta\boldsymbol{\theta} = \left\{[U']^{-1}[D']^{-1}[L']^{-1} - [U']^{-1}[D']^{-1}[L']^{-1}[M'][C'][M']^t[U']^{-1}[D']^{-1}[L']^{-1}\right\}\Delta\mathbf{P}$$

Agrupando os termos da equação acima de modo conveniente, chega-se à expressão (3.11):

$$\Delta\boldsymbol{\theta} = [U']^{-1}\left\{[I] - [D']^{-1}[L']^{-1}[M'][C']\underbrace{[M']^t[U']^{-1}}_{*}\right\}[D']^{-1}[L']^{-1}\Delta\mathbf{P} \qquad (3.11)$$

Onde,

$$[C'] = \left[[W'_B] + \underbrace{[M']^t[U']^{-1}}_{*}[D']^{-1}[L']^{-1}[M']\right]^{-1}$$

Em (3.11), $[I]$ é a matriz identidade de ordem $n \times n$.

Através de manipulações algébricas procura-se escrever a equação (3.11) no modo mais adequado à execução dos cálculos com matrizes. Nas deduções seguintes é suposto que a matriz $[B']$ é simétrica, portanto, $[U'] = [L']^t$ e $[U']^{-1} = [L']^{-t}$, onde t significa transposto. Por conveniência, define-se uma matriz $[\tilde{W}']$ como $[\tilde{W}'] \triangleq [D']^{-1}[L']^{-1}\,[M']$ (cuja ordem é $n \times q$).

Utilizando propriedades matriciais deseja-se expressar o termo indicado com $*$ em (3.11) como uma função da matriz $[\tilde{W}']$. A relação procurada está mostrada a seguir:

$$[M']^t[U']^{-1} = ([L']^{-1}\,[M'])^t = ([D']\,[\tilde{W}'])^t$$

Designando-se $[D']\,[\tilde{W}']$ por $[W']$ tem-se uma maneira alternativa de escrever a expressão (3.11), conforme mostra a equação (3.12).

$$\boldsymbol{\Delta\theta} = [L']^{-t}\left\{[D']^{-1}[L']^{-1}\boldsymbol{\Delta P} - [\tilde{W}']\,[C'][W']^t[D']^{-1}[L']^{-1}\boldsymbol{\Delta P}\right\} \tag{3.12}$$

Onde,

$$[C'] = \left[[W'_B] + [W']^t\,[\tilde{W}']\right]^{-1}$$

Quanto à magnitude da tensão, através de um procedimento análogo ao que se fez para ângulo obtém-se a expressão (3.13):

$$\boldsymbol{\Delta V} = [L'']^{-t}\left\{[D'']^{-1}[L'']^{-1}\boldsymbol{\Delta Q} - [\tilde{W}'']\,[C''][W'']^t[D'']^{-1}[L']^{-1}\boldsymbol{\Delta Q}\right\} \tag{3.13}$$

Onde,

$$[C''] = \left[[W''_B] + [W'']^t\,[\tilde{W}'']\right]^{-1}$$

As equações (3.12) e (3.13) estão no modo adequado para se efetuar os cálculos com matrizes e de matrizes com vetores, de acordo com o método *mid*-compensação [21].

Algoritmo *Mid*-compensação (Análise Completa)

Fase preparatória:

1. Resolver o sistema linear $[L'][D']\,[\tilde{W}'] = [M']$. Esse cálculo é efetuado por repetidas operações: em cada operação entra-se com o vetor independente $\mathbf{e}'_{ij}$. Cada retorno fornece como solução uma coluna da matriz $[\tilde{W}']$. De modo análogo, resolve-se o sistema linear $[L''][D'']\,[\tilde{W}''] = [M'']$;

2. Efetuar as seguintes multiplicações para obter as matrizes $[W']$ e $[W'']$:

$$[W'] = [D'][\tilde{W'}]$$

$$[W''] = [D''][\tilde{W''}]$$

3. Efetuar as multiplicações entre as matrizes auxiliares resultantes dos passos 1 e 2 (desta fase) para a obtenção das matrizes quadradas $(q \times q)$ de equivalentes, $[W'_T]$ e $[W''_T]$:

$$[W'_T] = [W']^t [\tilde{W'}]$$

$$[W''_T] = [W'']^t [\tilde{W''}]$$

4. Calcular as matrizes $[C']$ e $[C'']$, assim:

$$[C'] = \left[[W'_B] + [W'_T]\right]^{-1}$$

$$[C''] = \left[[W''_B] + [W''_T]\right]^{-1}$$

Estas operações podem ser feitas por intermédio de inversão de matrizes já que as matrizes envolvidas não são esparsas e têm ordem baixa;

5. Calcular os produtos matriciais $[\tilde{W'}]\,[C']$ e $[\tilde{W''}]\,[C'']$.

Fase de solução:

1. Calcular os vetores de *mismatch*, $\Delta\mathbf{P}$ e $\Delta\mathbf{Q}$;
2. Resolver os sistemas lineares $[L'][D']\,\hat{\mathbf{F}}' = \Delta\mathbf{P}$ e $[L''][D'']\,\hat{\mathbf{F}}'' = \Delta\mathbf{Q}$;
3. Calcular os vetores de compensação:

   1ª parte: Obter $[W']^t\,\hat{\mathbf{F}}'$ e $[W'']^t\,\hat{\mathbf{F}}''$

   2ª parte: Obter $\Delta\mathbf{F}' = [\tilde{W'}]\,[C'][W']^t\,\hat{\mathbf{F}}'$ e $\Delta\mathbf{F}'' = [\tilde{W''}]\,[C''][W'']^t\,\hat{\mathbf{F}}''$
4. Fazer a compensação, assim:

$$\mathbf{F}' = \hat{\mathbf{F}}' - \Delta\mathbf{F}'$$

$$\mathbf{F}'' = \hat{\mathbf{F}}'' - \Delta\mathbf{F}''$$

5. Resolver os sistemas lineares $[L']^t\Delta\theta = \mathbf{F}'$ e $[L'']^t\Delta\mathbf{V} = \mathbf{F}''$.
6. Atualizar, ao final de cada meia iteração (ativa e reativa), o respectivo vetor de estado:

$$\theta = \theta^0 + \Delta\theta$$

$$\mathbf{V} = \mathbf{V}^0 + \Delta\mathbf{V}$$

3.2.2 Atualização de Fatores

O ponto fundamental para a implementação das técnicas de vetores esparsos consiste em achar a árvore de fatoração para os fatores triangulares da matriz ($[L']$ ou $[U']$ ou $[L']^t$, por exemplo, se a matriz em foco for $[B']$).

De posse de um esquema de aproveitamento da esparsidade de vetores, conforme descrito na referência [22], a implementação do método de atualização de fatores fica trivial.

A Figura 3.1 ilustra o efeito da modificação parcial de fatores triangulares sobre as matrizes $[L']$, $[D']$ e $[L']^t$. Neste exemplo, em particular, supõe-se a alteração das impedâncias dos ramos $2-11$ ou $2-20$.

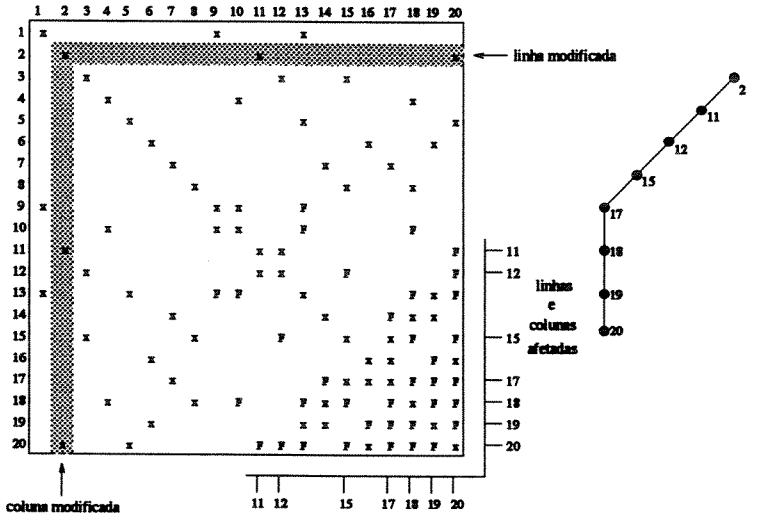


Figura 3.1: Caminho de fatoração e atualização de fatores para a matriz $[B']$.

Observa-se na Figura 3.1 que, a modificação dos fatores da linha/coluna 2 resultará na alteração de apenas aqueles fatores que se encontram nas linhas/colunas que fazem parte do caminho de fatoração para o nó 2. Conseqüentemente, o procedimento de atualização de fatores (ou refatoração parcial) fundamenta-se na árvore de fatoração da rede e no mecanismo de eliminação de Gauss. Estes são os mesmos fundamentos dos métodos de vetores esparsos.

Fazendo-se uma analogia entre fatoração e eliminação *forward* fica claro que, somente as colunas envolvidas no apropriado caminho serão atualizadas em $[L']$ quando a j-ésima linha/coluna de $[B']$ for modificada. (*Forward* é também conhecida como substituição direta; operação que tem lugar na solução de um sistema de equações quando a matriz de coeficientes está decomposta em seus fatores.)

Neste trabalho, com base em [24], foi implementada uma rotina de atualização de fatores que tem origem no método de Bennett. Para melhorar o desempenho das operações *fast-forward* e também da atualização de fatores, um método heurístico de ordenação é utilizado, denominado *MD-ML*. Esta heurística, baseada em simples manipulações da árvore de fatoração, minimiza localmente o comprimento dos caminhos de fatoração, enquanto preserva a esparsidade da matriz [28].

A técnica de atualização de fatores foi utilizada em diversas oportunidades neste trabalho. Um dos empregos é na análise de contingências de ordens maiores que quatro (por exemplo, na simulação de seis saídas simultâneas de ramos). Outro exemplo do emprego desta técnica foi nos cálculos de FPORS, sempre que se fez necessário trabalhar com a rede elétrica alterada (isto é, sob contingência; vide fluxograma 2.9) e a matriz $[B']$ precisa retratar esta condição.

Ao leitor interessado em conseguir mais informações sobre desenvolvimentos relacionados especificamente à esparsidade sugere-se o estudo das referências [1,4,22,24,28,29,65].

Finalmente, ressalta-se que os métodos baseados em compensação e a técnica de atualização de fatores podem também ser usados na obtenção de soluções mais rápidas que as soluções completas, porém aproximadas. Para isso, basta apenas efetuar uma meia iteração (i.e., $1P\theta$) para obter o estado angular aproximado, ou efetuar uma iteração completa (i.e., $1P\theta-1QV$) para determinar os ângulos e as magnitudes aproximados das tensões nas barras. Desse modo, tais técnicas podem também ser usadas como métodos de *screening* de contingências. Além dessas alternativas, a seção 3.2.3 descreve métodos de *screening* especializados na obtenção rápida de estados pós-contingências.

3.2.3 Seleção de Contingências: Métodos de *Screening*

As técnicas verdadeiramente completas de análise de contingências devem considerar o impacto efetivo sobre as tensões e sobre os fluxos de potência nos ramos. Para fazer isto com rigor exige-se um fluxo de potência AC. Modernas técnicas empregam o desacoplado rápido associado ao Lema de Inversão de Matrizes. Embora esses programas sejam bastante rápidos, os tempos de computação são ainda elevados para se aplicar uma completa análise AC a um grande número de contingências. Este dilema é resolvido encontrando-se uma maneira de limitar ou "peneirar" os casos que devem ser examinados através de análise de contingências AC. Tal procedimento é conhecido na literatura como *screening* [27].

Screening é uma forma de seleção de contingências; outros métodos (conhecidos como indiretos) baseiam-se na produção de índices de severidade dos casos de contingência para subseqüente classificação, sem, no entanto, calcular as quantidades pós-contingentes monitoradas (as técnicas de fatores de distribuição pertencem à categoria dos métodos indiretos) [14,16,17,19].

Muitas especificações para Centros de Supervisão e Controle envolvem alguma forma de

análise linear como parte integrante da aplicação computacional a ser usada na determinação da segurança do sistema. Uma combinação prática, que normalmente funciona bem, consiste da análise completa de contingências baseada no Lema de Inversão de Matrizes, e de um tipo de análise linear de contingências (*DC* ou *DC* incremental, ou uma iteração do desacoplado rápido). Nesta combinação, a análise linear de contingências serve como uma função de pré-processamento para examinar todas as contingências de uma lista inicial ("bruta", original). Aqueles casos de contingência com as piores violações são então adicionados à lista de contingências a ser processada com mais detalhes através de uma análise exata.

Como resultado final do pré-processamento espera-se obter uma lista reduzida com o *ranking* das contingências e, além disso, uma relação de casos que conduzem a ilhamentos do sistema.

A) *Screening* pelo Método *DC* Incremental

Este método usa superposição linear e trata contingências simples e múltiplas, possibilitando uma rápida e confiável obtenção dos ângulos das tensões nodais pós-contingência. As referências pesquisadas e que deram origem a este método são [5] e [19].

Para este tipo de análise linearizada, um modelo incremental para mudanças em torno de um ponto de operação é mais apropriado. Assumindo-se que uma solução do fluxo de potência do caso-base tenha sido obtida, mudanças a partir deste ponto de operação podem ser expressas pela relação (3.14).

$$\mathbf{\Delta P} = [B'] \, \mathbf{\Delta \theta} \tag{3.14}$$

Onde,

$\mathbf{\Delta P}$ é o vetor de correções de injeções de potência ativa, $(n \times 1)$;

$\mathbf{\Delta \theta}$ é o vetor de mudanças de ângulos das tensões nodais, $(n \times 1)$;

$[B']$ é a matriz $(n \times n)$ definida em (2.6), no capítulo anterior.

Os fluxos incrementais de ramos são calculados como mostra a equação (3.15).

$$\Delta P_{km} = \frac{\Delta \theta_k - \Delta \theta_m}{x_{km}} = x_{km}^{-1} \Delta \theta_{km} \quad \text{ou} \quad \Delta P_{km} = -b_{km} \Delta \theta_{km} \tag{3.15}$$

O desenvolvimento deste método fundamenta-se no Teorema da Compensação (também conhecido como Teorema da Substituição) brilhantemente explanado por Tinney em [5]. Devido à exigüidade de espaço, os detalhes das deduções para se chegar ao algoritmo do método *DC* incremental não são fornecidos [60].

Screening DC Incremental para Contingência Simples

Deseja-se remover um ramo que conecta os nós k e m pelo qual passa o fluxo ativo P_{km}^0. Seja z_{km} a impedância de um novo ramo temporariamente ligado entre k e m, em paralelo com

os ramos já existentes na rede. Para $z_{km} = -x_{km}$ (ou $z_{km} = 1/b_{km}$), o efeito é o mesmo que abrir o ramo original.

A remoção do ramo de impedância x_{km} é efetivamente consumada redistribuindo-se o fluxo pré-contingência, P^0_{km}, através dos ramos remanescentes. A variação total de injeção é P^0_{km}.

Obtenção de w_T (impedância equivalente de Thevenin):

A expressão para w_T pode ser escrita em função do vetor $\mathbf{e}_{km}$, do seu transposto ($\mathbf{e}^t_{km}$), e da matriz $[B']$:

$$\begin{aligned} w_T &= \mathbf{e}^t_{km}[B']^{-1}\mathbf{e}_{km} \\ &= \mathbf{e}^t_{km}\mathbf{w}_{km} \end{aligned}$$

Conhecidos os fatores triangulares $[L']$, $[D']$ e $[L']^t$ de $[B']$, o vetor $\mathbf{w}_{km}$ pode ser obtido resolvendo-se o sistema linear (3.16):

$$[B']\mathbf{w}_{km} = \mathbf{e}_{km} \tag{3.16}$$

Por fim, w_T é determinada subtraindo-se o conteúdo das posições k e m do vetor $\mathbf{w}_{km}$.

Com base no Teorema da Compensação é possível chegar à expressão da variação do estado produzida pela retirada de x_{km} em função do caso-base e de parâmetros da rede.

$$\Delta\theta = -\frac{x_{km}P^0_{km}}{w_T - x_{km}}\mathbf{w}_{km} \tag{3.17}$$

Diante do que foi apresentado, pode-se resumir o método *DC* incremental para remoção de um único ramo da rede no seguinte algoritmo.

Algoritmo *DC* Incremental para Contingência Simples

1. Obter o estado do caso-base (θ^0, $\mathbf{V}^0$) e os fluxos ativos *AC* dos ramos da rede, resolvendo-se um fluxo de potência;
2. Definir o ramo $k-m$ a ser removido e tomar o seu fluxo ativo pré-contingência, P^0_{km};
3. Resolver o sistema linear $[B']\mathbf{w}_{km} = \mathbf{e}_{km}$;
4. Calcular w_T com base no resultado do passo 3;
5. Calcular o novo estado θ por meio da expressão:

$$\theta = \theta^0 + \Delta\theta = \theta^0 - \frac{x_{km}P^0_{km}}{w_T - x_{km}}\mathbf{w}_{km}$$

6. Voltar ao passo 2 até exaurir a lista de contingências.

Screening DC Incremental para Contingência Múltipla

Nesta seção, o método *DC* incremental, aplicado anteriormente à análise de contingências simples, é estendido ao estudo de contingências múltiplas. É estudada a situação em que ocorrem simultaneamente as variações $z_{k_l m_l}$ nas impedâncias das ligações $k_l - m_l$ ($l = 1, 2, \cdots, q$), provocadas por uma contingência múltipla. Neste tipo de contingência, várias linhas/transformadores são removidas ao mesmo tempo da rede de transmissão. O procedimento simultâneo em foco é aplicável quando se conhece *a priori* as contingências simples que constituem a contingência múltipla.

Considera-se inicialmente a saída simultânea de dois ramos $k - m$ e $i - j$. Isto equivale a adicionar os elementos z_{km} e z_{ij} à rede intacta. Os fluxos ativos pré-contingência são P^0_{km} e P^0_{ij}.

O procedimento é análogo ao que foi feito para contingência simples. Todavia, para contingência múltipla, $[W_T]$ é uma matriz de ordem q e tem origem na relação $w_T = \mathbf{e}^t_{km}[B']^{-1}\mathbf{e}_{km}$, a qual pode ser reescrita como a equação matricial (3.18).

$$[W_T] = [M]^t\,[B']^{-1}\,[M] \tag{3.18}$$

$$= \begin{array}{|c|c|}\hline \mathbf{e}_{km} & \mathbf{e}_{ij} \\ \hline \end{array}^t [B'] \begin{array}{|c|c|}\hline \mathbf{e}_{km} & \mathbf{e}_{ij} \\ \hline \end{array}$$

$$= \begin{array}{|c|c|}\hline \mathbf{e}_{km} & \mathbf{e}_{ij} \\ \hline \end{array}^t \begin{array}{|c|c|}\hline \mathbf{w}_{km} & \mathbf{w}_{ij} \\ \hline \end{array}$$

Seja $[W_B]$ a matriz diagonal que contém as alterações de impedâncias de ramo:

$$[W_B] = \begin{array}{|c|c|}\hline z_{km} & \\ \hline & z_{ij} \\ \hline \end{array}$$

A partir do Teorema da Compensação é fácil provar que as variações de injeções que penetram a rede intacta são como mostra a equação (3.19).

$$\begin{array}{|c|}\hline \Delta P_{km} \\ \hline \Delta P_{ij} \\ \hline \end{array} = \{[W_T] + [W_B]\}^{-1}\,[W_B] \begin{array}{|c|}\hline P^0_{km} \\ \hline P^0_{ij} \\ \hline \end{array} \tag{3.19}$$

Sabendo-se que $[B']^{-1}\mathbf{e}_{km} = \mathbf{w}_{km}$ e $[B']^{-1}\mathbf{e}_{ij} = \mathbf{w}_{ij}$, finalmente, a obtenção do vetor $\Delta\theta$ é conseguida pela aplicação da expressão (3.20):

$$\Delta\theta = \Delta P_{km}\mathbf{w}_{km} + \Delta P_{ij}\mathbf{w}_{ij} \tag{3.20}$$

As expressões mostradas nesta seção podem ser perfeitamente generalizadas para contingências de ordem q (para $q = 1, 2, 3, \ldots$). Para q saídas simultâneas de ramos, $[W_T]$ torna-se uma matriz $q \times q$ e $[W_B]$ uma matriz diagonal $q \times q$. A inversão de $\{[W_T] + [W_B]\}$ permanece trivial para um valor de q razoavelmente pequeno (por exemplo, no máximo igual a seis). A matriz $[W_T]$ de impedância equivalente é obtida de modo análogo ao que foi explicado para contingência simples, ou seja, através da solução de q sistemas lineares do tipo mostrado em (3.16).

Portanto, para q contingências simultâneas de ramos $k_l - m_l$ ($l = 1, 2, \cdots, q$) aplica-se a expressão geral (3.21):

$$\Delta\theta = \Delta P_{k_1m_1}\mathbf{w}_{k_1m_1} + \Delta P_{k_2m_2}\mathbf{w}_{k_2m_2} + \ldots + \Delta P_{k_qm_q}\mathbf{w}_{k_qm_q} \tag{3.21}$$

Diante do que foi apresentado, pode-se resumir o método *DC* incremental para remoção simultânea de q ramos da rede no seguinte algoritmo.

Algoritmo *DC* Incremental para Contingência Múltipla

1. Obter o estado do caso-base (θ^0, $\mathbf{V}^0$) e os fluxos ativos AC dos ramos da rede, resolvendo-se um fluxo de potência;
2. Definir os ramos $k_l - m_l$ ($l = 1, 2, \cdots, q$) a serem removidos e tomar seus fluxos AC, $P^0_{k_lm_l}$ ($l = 1, 2, \cdots, q$);
3. Obter a matriz diagonal $[W_B]$ a partir do passo 2;
4. Resolver os sistemas lineares esparsos $[B']\mathbf{w}_{k_lm_l} = \mathbf{e}_{k_lm_l}$, com $l = 1$ até q;
5. Obter a matriz $[W_T]$ com base no resultado do passo 4;
6. Calcular por uma expressão análoga à equação (3.19) as variações de fluxos $\Delta P_{k_lm_l}$, ($l = 1, 2, \cdots, q$);
7. Calcular o novo estado θ por meio da expressão:

$$\theta = \theta^0 + \Delta P_{k_1m_1}\mathbf{w}_{k_1m_1} + \Delta P_{k_2m_2}\mathbf{w}_{k_2m_2} + \ldots + \Delta P_{k_qm_q}\mathbf{w}_{k_qm_q}$$

8. Voltar ao passo 2 até exaurir a lista de contingências.

B) *Screening* $1P\theta - 1QV$

O cálculo de *screening* $1P\theta - 1QV$, de acordo com a abordagem adotada nos programas desta tese, consiste na determinação dos estados pós-contingências, θ e $\mathbf{V}$, aproximados da rede elétrica. De posse dos incrementos $\mathbf{\Delta\theta}$ e $\mathbf{\Delta V}$, estes são adicionados ao estado básico obtido previamente de um fluxo de potência. Em seguida, é elaborada a lista com o *ranking* das perturbações simuladas. Geralmente, as informações do *screening* $1P\theta - 1QV$ possibilitam a obtenção de uma classificação das contingências em função de desvios da tensão nas barras de carga ou das sobrecargas de potência aparente nos ramos [17].

Este método equivale, na verdade, à primeira iteração completa do fluxo de potência desacoplado rápido. Em especial, este método foi implementado para o propósito de comparações dos métodos diretos e iterativos de solução de sistemas $[A]\mathbf{x} = \mathbf{b}$ aplicados à análise de contingências (vide Apêndice E).

A contingência é simulada compensando-se a saída do ramo através da consideração da igualdade entre as injeções incrementais nas barras terminais e o fluxo pré-contingência no ramo, ou seja, no caso-base. A Figura 3.2 ilustra este princípio.

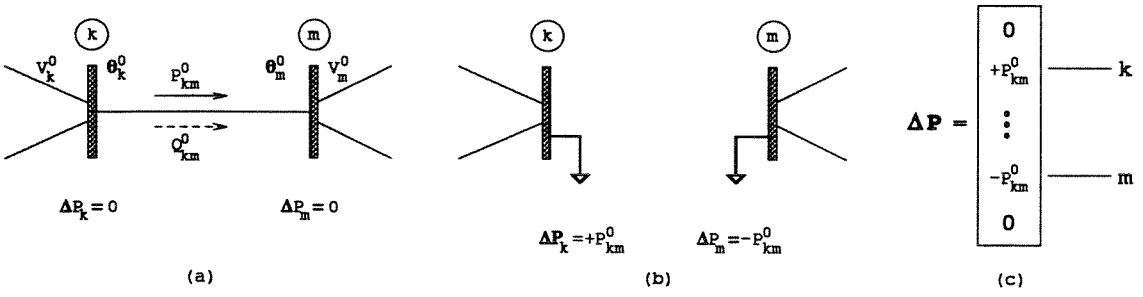


Figura 3.2: Princípio utilizado no cálculo de *screening*: (a) rede no caso-base (fluxo de potência convergido); (b) rede pós-contingência com injeções de potência; (c) vetor de *mismatch* pós-contingência.

Algoritmo $1P\theta - 1QV$

Conhecidos o estado $\mathbf{V}^0$ e θ^0 e os fatores das matrizes $[B']$ e $[B'']$ do caso-base, o algoritmo de *screening* $1P\theta - 1QV$ compõe-se dos seguintes passos.

1. Calcular fluxos ativos e reativos nos ramos para o caso-base;
2. Retirar o ramo $k - m$, cujos fluxos do caso-base são P_{km}^0 e Q_{km}^0;

3. Resolver $[B']\Delta\theta = \Delta\mathbf{P}$, onde $\Delta\mathbf{P} = \Delta\mathbf{P}(\mathbf{V}^0, \theta^0)$ e $[B']$ deve refletir a alteração. Neste passo, utiliza-se a equação (3.1);
4. Atualizar θ: $\theta \leftarrow \theta^0 + \Delta\theta$;
5. Compensar o efeito de se ter $\Delta\mathbf{Q} = \Delta\mathbf{Q}(\mathbf{V}^0, \theta^0) = (0 \cdots + Q^0_{km} \cdots - Q^0_{km} \cdots 0)^t$, e de se precisar de $\Delta\mathbf{Q}(\mathbf{V}^0, \theta)$ com θ pós-contingência;
6. Resolver $[B'']\Delta\mathbf{V} = \Delta\mathbf{Q}$, onde $\Delta\mathbf{Q} = \Delta\mathbf{Q}(\mathbf{V}^0, \theta)$ e $[B'']$ deve refletir a alteração. Neste passo, utiliza-se a equação (3.2);
7. Atualizar $\mathbf{V}$: $\mathbf{V} \leftarrow \mathbf{V}^0 + \Delta\mathbf{V}$;
8. Voltar ao passo 2 até exaurir a lista de contingências.

Conforme o artifício ilustrado na Figura 3.2, o vetor de *mismatch* (ativo ou reativo) possuirá apenas duas posições não-nulas no caso de contingência simples, portanto, as soluções dos passos 3 e 6 podem ser obtidas utilizando-se métodos de vetores esparsos. Os cálculos destes passos podem ser efetuados utilizando-se pós-compensação ou *mid*-compensação, descritas anteriormente. Quando é efetuada apenas a primeira meia iteração (i.e., ativa) tem-se o chamado *screening* $1P\theta$.

Compensação referida no passo 5:

Como no passo 1 foram obtidos os fluxos em todos os ramos da rede para o estado básico $(\theta^0, \mathbf{V}^0)$, ao processar o passo 6 deve-se utilizar o *mismatch* $\Delta\mathbf{Q}$ compensado, uma vez que já se tem nesta fase o valor atualizado de θ. Das equações incrementais do fluxo de potência extrai-se a compensação deste efeito.

$$\Delta\mathbf{Q}(\mathbf{V}^0, \theta^0) - \left[\frac{\partial\mathbf{Q}}{\partial\theta}\right]\Delta\theta = [B'']\Delta\mathbf{V} \tag{3.22}$$

A parcela $\Delta\mathbf{Q}(\mathbf{V}^0, \theta^0)$ já é conhecida do passo 1 (fluxos reativos do caso-base). A conclusão realmente importante é que, a equação (3.22) é equivalente à equação (3.23).

$$\Delta\mathbf{Q}(\mathbf{V}^0, \theta^0 + \Delta\theta) = [B'']\Delta\mathbf{V} \tag{3.23}$$

Neste trabalho, o produto $\left[\frac{\partial\mathbf{Q}}{\partial\theta}\right]\Delta\theta$ foi aproximado pelo produto $[G]\Delta\theta$, onde $[G]$ é a parte real da matriz admitância nodal, $[Y]$. Testes de validação desta compensação para o *screening* $1P\theta - 1QV$ foram feitos comparando-se com resultados de rotinas de cálculo exato (por exemplo, de uma iteração completa do desacoplado rápido), e a precisão dos resultados mostrou-se satisfatória.

3.2.4 Formação do *Ranking* das Contingências

As técnicas de análise de contingências estudadas até este ponto servem para determinar o estado da rede $(\theta, \mathbf{V})$ pós-distúrbio, seja de modo exato ou aproximado.

Monitorando as apropriadas quantidades pós-contingência (e.g., fluxos em linhas, tensões em barras de carga, etc.), a severidade do caso simulado pode ser quantificada diretamente por meio de algum tipo de heurística com a finalidade de formar um *ranking*. A medida da severidade de um caso é freqüentemente um único número: o índice de *performance*, PI.

O objetivo desta seção consiste tão somente em apresentar os índices utilizados na fase de seleção de contingências dos diferentes programas implementados.

Índice de *Performance* para o *Ranking* de $[MW]$

Nos programas de FPORS, o índice de *performance* utilizado é adequado ao subproblema ativo, onde o interesse reside em "filtrar" os casos de contingências que levam à sobrecargas de fluxos ativos em ramos do sistema. É o índice usual na formação do *ranking* de MW, simbolizado por PI_{MW}.

Este índice (PI_{MW}) foi introduzido por Ejebe e Wollenberg [11] e foi posteriormente modificado [14,30]. A definição (3.24) mostra a fórmula do índice PI_{MW} empregado:

$$PI^{\nu}_{MW} = \sum_{ij \in S_r} \left(\frac{P^{\nu}_{ij}}{P^{\lim}_{ij}} \right)^2 \tag{3.24}$$

Onde,

P^{ν}_{ij} é o fluxo no ramo $i-j$ após a contingência ν;

$P^{\lim}_{ij}$ representa o limite de fluxo no ramo $i-j$ (i.e., valores superior e inferior especificados, respectivamente, $P^{\text{máx}}_{ij}$ e $P^{\text{mín}}_{ij}$);

S_r é o conjunto de ramos em sobrecarga, ou seja, somente aqueles ramos cujos limites foram ultrapassados no estado pós-contingência.

De posse do vetor θ^{ν} das barras da rede, resultante da simulação do caso ν, é fácil avaliar os fluxos nos ramos remanescentes pela expressão linearizada $P^{\nu}_{ij} = -b_{ij}\theta^{\nu}_{ij}$. O conjunto S_r será então formado verificando-se qual ramo teve um dos limites violado. A idéia por traz do índice é ter um escalar associado a cada contingência, cujo valor seja pequeno quando os ramos estiverem sobrecarregados próximos dos seus limites, e o valor seja alto (i.e., grande) quando os limites estiverem severamente violados. Ramos não sobrecarregados não devem contribuir para aumentar o valor do índice.

O índice PI_{MW} foi originalmente definido com as seguintes características:

- Somatório sobre todos os ramos do sistema, sobrecarregados ou não;
- Expoente $2n$ em lugar de simplesmente 2 (n é um número inteiro);
- Fator de ponderação para cada ramo e normalizado por $2n$.

A soma sobre todos os ramos é susceptível ao fenômeno de mascaramento, onde uma contingência que leva a um número de linhas pesadamente carregadas, porém nenhuma sobrecarga, é classificada tal qual uma contingência que produz uma única linha sobrecarregada. O expoente $2n$ para $n = 1$ é preferido, em virtude da rapidez dos cálculos se comparada a outros valores de n. Entretanto, quando $n = 1$, podem ocorrer situações em que a habilidade do índice PI_{MW} para distinguir ou detectar casos severos é limitada. O efeito aparece na ordenação das contingências pelos valores PI_{MW}, podendo resultar numa lista na qual nem todos os casos representativos, tidos como realmente severos, figurem no topo. O fator de ponderação pode ser usado para refletir a importância de determinadas linhas ou ramos do sistema.

Índice de *Performance* para o *Screening* $1P\theta - 1QV$

Os métodos de *screening* que são capazes de fornecer resultados sobre grandezas elétricas associadas ao subproblema reativo, por exemplo, o *screening* $1P\theta - 1QV$, possibilitam classificar as contingências também quanto à violações de limites das tensões em barras PQ ou das injeções de potências reativas. Há diversos índices que podem ser empregados para quantificar o efeito das contingências sobre variáveis reativas. Neste trabalho, apenas nos programas de seleção de contingências elaborados exclusivamente para estudar a aplicabilidade dos métodos iterativos (e.g., gradiente conjugado pré-condicionado) na solução de sistemas algébricos lineares esparsos é que foi utilizado um índice para tensões (vide Apêndice E).

O índice utilizado no *screening* $1P\theta - 1QV$ é simbolizado por PI_v e definido por (3.25) [44].

$$PI_v^{\nu} = \sum_{i \in PQ} \left(\frac{V_i^{\lim} - V_i^{\nu}}{V_i^{\lim}} \right)^n \tag{3.25}$$

Onde, V_i^{ν} é a tensão na barra i após a contingência ν, e a expressão é definida de modo que a soma é tomada apenas sobre as barras PQ onde ocorram violações das magnitudes de tensão; n é um inteiro par.

Na expressão (3.25), $V_i^{\lim}$ representa o limite superior ou inferior de segurança para a magnitude de tensão na barra i. Os valores para os limites superior $V_i^{\text{máx}}$ e inferior $V_i^{\text{mín}}$ são arbitrados a partir das magnitudes das tensões do caso-base conhecidas, ou seja, são obtidos através das seguintes expressões:

$$V_i^{\text{máx}} = V_i^0 + 0,05 \qquad [pu]$$

$$V_i^{\text{mín}} = V_i^0 - 0,05 \qquad [pu]$$

Conforme está documentado em [44], a margem $0,05$ $[pu]$ para a determinação dos limites foi adotada após simulações exaustivas com muitos outros valores; o valor de n adotado foi 4.

Na formação de *ranking* das contingências, principalmente quando se deseja verificar a eficácia de um índice e a qualidade de um método de *screening*, uma medida da exatidão da filtragem é usual. A porcentagem da exatidão da filtragem de um método de obtenção de *ranking* é definida por (3.26) [30].

$$\% \text{ da exatidão da filtragem } = \frac{N_c}{N_{c,ref}} \times 100 \tag{3.26}$$

Onde, $N_{c,ref}$ é o número de casos críticos de contingências obtidos pelo método tomado como referencial, e N_c é o número de casos críticos de contingências identificados pelo método em teste. Esta medida indica quão corretamente um método discrimina entre os casos críticos e não críticos de contingências no conjunto de todas as contingências postuladas.

3.2.5 Como os Métodos de Análise de Contingências Foram Utilizados?

As técnicas de análise de contingências descritas nas seções precedentes foram implementadas para execução em computador digital, compondo rotinas especializadas.

Nos programas de FPORS apresentados no capítulo anterior, as técnicas de análise de contingências foram utilizadas em quatro diferentes oportunidades:

1. Obtenção do *ranking* das contingências em relação às sobrecargas de fluxos ativos em ramos, tal como ilustra o diagrama de blocos da Figura 2.5, onde está mostrado o *loop* de 1 a N_c. Nesta aplicação foram empregados os algoritmos do método *DC* incremental descritos na seção 3.2.3. Foram também implementados os métodos de pós-compensação, *mid*-compensação e atualização de fatores para a meia iteração ativa.

2. Cálculo do estado θ pós-contingência à medida que se avança na solução do FPORS, tal como ilustra os diagramas de blocos das Figuras 2.6, 2.7 e 2.9. Este processamento da análise de contingência tem a finalidade de identificar violações nos estados pós-contingência para cada novo ponto de operação que é obtido quando se "caminha" na direção da solução ótima-segura. Nesta aplicação foi utilizado o método *DC* incremental.

   Os métodos empregados nas oportunidades 1 e 2 foram utilizados apenas para resolver a meia iteração do fluxo de potência desacoplado rápido ($1P\theta$).

3. Alteração da matriz $[B']$ da rede elétrica por atualização de fatores triangulares em função da contingência para resolver os subproblemas de operação no programa FPORS-III, conforme ilustra o fluxograma da Figura 2.9.

4. Obtenção das restrições de contingências, conforme a abordagem de Stott, construindo-se a restrição para a máxima violação para os programas FPORS-I e FPORS-II. A seção 3.3 a seguir descreve a utilização das técnicas de análise de contingências na obtenção dessas restrições.

Visando minimizar o tempo global de processamento da análise de contingências, sempre que possível, a abordagem adotada nas aplicações é seletiva: como são tratadas no programa tanto contingências simples quanto múltiplas, ou seja, a lista é em geral mista, utiliza-se pós-compensação, *mid*-compensação ou atualização de fatores, dependendo da ordem da contingência (aqui designada por q). A Figura 3.3 ilustra como é feita esta seletividade nos cálculos.

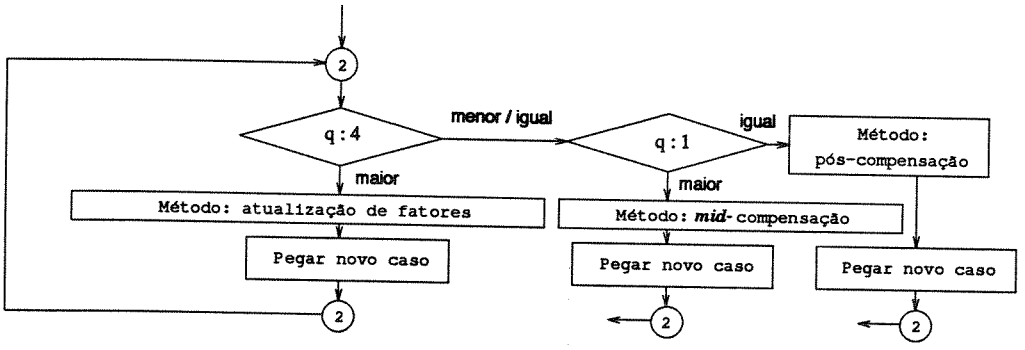


Figura 3.3: Seletividade nos cálculos das contingências simples e múltiplas.

Nos programas de *screening* de contingências, elaborados exclusivamente com o propósito de fazer comparações entre os métodos iterativos e diretos de solução de $[A]\mathbf{x} = \mathbf{b}$, foram empregados todos os algoritmos descritos anteriormente. Quanto aos métodos iterativos, estes foram aplicados aos algoritmos $1P\theta$ e $1P\theta - 1QV$. Ambos os algoritmos requerem as soluções dos sistemas $[B']\Delta\theta = \Delta\mathbf{P}$ e/ou $[B'']\Delta\mathbf{V} = \Delta\mathbf{Q}$, que são suas operações matemáticas mais "pesadas" computacionalmente. Em soluções desta natureza é que são aplicados os métodos baseados em fatoração LDU (designados por diretos) ou os métodos iterativos do gradiente conjugado pré-condicionado. O leitor é convidado a reportar-se ao Apêndice E, onde são feitas considerações e avaliações comparativas a respeito deste interessante assunto.

3.3 Restrições de Contingências para o Fluxo de Potência Ótimo

Neste trabalho, dois tipos de restrições de segurança são utilizadas nos programas de FPORS. Em primeiro lugar, tem-se a restrição obtida a partir do efeito de uma contingência sobre os ramos vizinhos, causando violações de seus limites. Este tipo de restrição é usada nos programas FPORS-I e FPORS-II, e é descrita em [9,10,12,13]. A outra categoria de restrição é

fundamentada na decomposição de Benders e utiliza multiplicadores simplex. Este último tipo é usada no programa FPORS-III e foi discutida no Capítulo 2 e nos Apêndices B e D [23,26].

Nesta seção, a restrição do tipo Stott (violação decorrente da saída de ramo) tem o seu princípio físico ilustrado e, em seguida, é feito o equacionamento para as situações de contingências simples e múltiplas.

A saída do ramo $i-j$ causa violações dos limites de fluxos dos ramos vizinhos. Suponha que o ramo que teve o limite violado seja $k-m$. A Figura 3.4 ilustra este fenômeno.

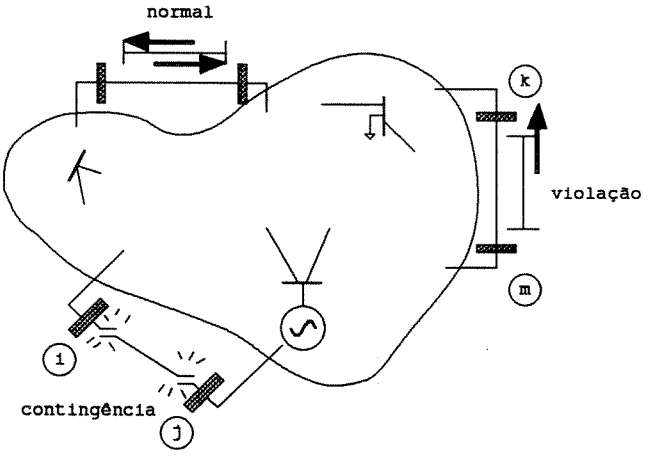


Figura 3.4: Efeito da contingência $i-j$ na região circunvizinha.

O objetivo é obter uma fórmula para calcular o fluxo incremental, Δf_{km}, num ramo qualquer remanescente, $k-m$. A expressão de Δf_{km} deve ficar em função de $\mathbf{\Delta P}$, que é o vetor das variáveis de decisão do processo de otimização, tal como indica (3.27).

$$\Delta f_{km} = \mathbf{A}_{km}^{t} \mathbf{\Delta P} \tag{3.27}$$

O fluxo incremental contingente no ramo que sofreu violação pode ser escrito em termos do estado $\mathbf{\Delta\theta}$ pós-contingência, assim: $\Delta f_{km} = -b_{km}(\mathbf{e}'_{km})^t \mathbf{\Delta\theta} = -b_{km}(\Delta\theta_k - \Delta\theta_m)$. Então, basta expressar o estado pós-contingência em termos do estado pré-contingência e do vetor $\mathbf{\Delta P}$. Para tanto, é só usar o modelo incremental da rede:

$$[B']\mathbf{\Delta\theta} = \mathbf{\Delta P} \rightarrow \mathbf{\Delta\theta} = [B']^{-1}\mathbf{\Delta P}$$

e, em seguida, aplicar o Lema de Inversão de Matrizes (vide equação (3.1)):

$$[B']^{-1} - [B']^{-1}\,\mathbf{e}'_{ij}\,(c')\,(\mathbf{e}'_{ij})^t\,[B']^{-1} \rightarrow \text{simples}$$

$$[B']^{-1} - [B']^{-1}\,[M']\,[C']\,[M']^t\,[B']^{-1} \rightarrow \text{múltipla}$$

A seguir, são estabelecidas as fórmulas das restrições por violação máxima para contingências simples e múltiplas, e a seqüência correta de cálculo em cada caso.

Restrição para Contingência Simples

$$\mathbf{A}_{km} = -b_{km}[B']^{-1}\left(\mathbf{e}'_{km} - \alpha_{km}\mathbf{e}'_{ij}\right) \tag{3.28}$$

Onde,

$$\alpha_{km} = (\mathbf{e}'_{km})^t[B']^{-1}\mathbf{e}'_{ij}c'$$

$$c' = \frac{1}{\frac{1}{b_{ij}} + (\mathbf{e}'_{ij})^t[B']^{-1}\mathbf{e}'_{ij}}$$

A seqüência correta de cálculo dos coeficientes da restrição $\mathbf{A}^t_{km}$, dada pela fórmula (3.28), é a seguinte (sempre que for possível, utilizar os métodos de vetores esparsos):

1. Resolver o sistema linear:

$$[B']\mathbf{w}^a = \mathbf{e}'_{ij}$$

2. De posse do resultado $\mathbf{w}^a$ e do parâmetro b_{ij} do ramo que saiu, calcular o escalar c', assim:

$$c' = \frac{1}{\frac{1}{b_{ij}} + \mathbf{w}^a(i) - \mathbf{w}^a(j)}$$

3. Calcular o escalar α_{km}, assim:

$$\alpha_{km} = (\mathbf{w}^a(k) - \mathbf{w}^a(m))c'$$

4. Resolver o sistema linear:

$$[B']\mathbf{w}^b = \mathbf{e}'_{km}$$

5. Avaliar o vetor $\mathbf{A}_{km}$ a partir dos valores calculados nos passos anteriores, onde o elemento da posição l é dado por:

$$\mathbf{A}_{km}(l) = -b_{km}\left[\mathbf{w}^b(l) - \alpha_{km}\mathbf{w}^a(l)\right]$$

Os passos iniciais deste algoritmo apresentam operações em comum com os passos do algoritmo pós-compensação dado na seção 3.2.1. Isto foi aproveitado durante a elaboração dos programas.

Restrição para Contingência Múltipla

$$\mathbf{A}_{km} = -b_{km}[B']^{-1}\left(\mathbf{e}'_{km} - [M']\mathbf{v}_{\alpha_{km}}\right) \tag{3.29}$$

Onde,

$$\mathbf{v}^t_{\alpha_{km}} = (\mathbf{e}'_{km})^t[B']^{-1}[M'][C']$$

$$[C'] = \left([W'_B] + [M']^t[B']^{-1}[M']\right)^{-1}$$

$$\mathbf{v}^t_{\alpha_{km}} = \begin{array}{|c|c|c|c|c|c|}\hline \alpha^1_{km} & \alpha^2_{km} & \cdots & \alpha^l_{km} & \cdots & \alpha^q_{km} \\ \hline \end{array}$$

Considerando-se a contingência de q ramos, $i_l - j_l$, $l = 1, 2, \cdots, q$, a seqüência correta de cálculo dos coeficientes da restrição $\mathbf{A}^t_{km}$, dada pela fórmula (3.29), é a seguinte:

1. Resolver o sistema matricial $[B'][W'] = [M']$, que corresponde a resolver q sistemas do tipo $[B']\mathbf{w}^a_{i_l j_l} = \mathbf{e}'_{i_l j_l}$ análogos ao do passo 1 do algoritmo anterior. O resultado dos cálculos deste passo é a matriz $[W']$;
2. A partir do resultado do passo 1 (acima), calcular o produto $[M']^t[B']^{-1}[M']$. Conhecida a matriz $[W']$, este cálculo é feito simplesmente subtraindo-se o conteúdo das posições correspondentes a i_l e j_l, para $l = 1, 2, \cdots, q$;
3. Calcular a matriz $[C']$ a partir da matriz diagonal $[W'_B]$ e do resultado do passo anterior;
4. De posse dos resultados dos passos 1, 2 e 3, calcular o vetor $\mathbf{v}^t_{\alpha_{km}}$:

$$\mathbf{v}_{\alpha_{km}} = (\mathbf{e}'_{km})^t[W'][C']$$

5. Calcular o produto $[W']\mathbf{v}_{\alpha_{km}} (= [B']^{-1}[M']\mathbf{v}_{\alpha_{km}})$ e armazenar o resultado no vetor $\mathbf{w}^c$. Portanto,

$$\mathbf{w}^c = [W']\mathbf{v}_{\alpha_{km}}$$

6. Resolver o sistema linear:

$$[B']\mathbf{w}^b = \mathbf{e}'_{km}$$

7. Avaliar o vetor $\mathbf{A}_{km}$ a partir dos valores calculados nos passos anteriores, onde o elemento da posição l é dado por:

$$\mathbf{A}_{km}(l) = -b_{km}\left[\mathbf{w}^b(l) - \mathbf{w}^c(l)\right]$$

3.4 Proposta de Obtenção de um *Ranking* de Contingências Baseado na Capacidade Corretiva do Sistema

Após a seleção de contingências tem-se uma lista ordenada dos casos, classificados em ordem decrescente da severidade das sobrecargas causadas no sistema. Este *ranking* é obtido a partir do valor do índice PI_{MW} associado a cada contingência simulada.

Uma vez identificadas as contingências graves de uma lista, resta planejar as ações corretivas/preventivas que eliminem os efeitos dos distúrbios. Este estudo é designado aqui como a determinação de estratégias corretivas e preventivas de sobrecargas. Nos mais modernos Centros de Supervisão e Controle do mundo, o programa de estratégia corretiva usa um algoritmo dual-simplex de programação linear para calcular os ajustes dos transformadores defasadores e dos geradores, para, com isso, determinar onde deve haver cortes de carga se necessário for (no caso do sistema estar inseguro), se houver linhas sobrecarregadas além dos limites normais ou de emergência. O processo de análise de contingências pode armazenar informações sobre cada contingência com sobrecarga, e pode, então, passar esta informação ao algoritmo de estratégia corretiva.

Nesta seção, considerando-se o que foi estabelecido anteriormente sobre análise de contingências e capacidade corretiva do sistema, é sugerida a utilização do algoritmo de solução de PL de Stott para a determinação de uma estratégia corretiva de sobrecargas, que seria executada após o conhecimento das contingências mais severas. O programa então implementado consiste das etapas preliminares ilustradas na Figura 2.5 associadas ao cálculo do FPORS onde é admitida flexibilidade máxima ($\Delta \rightarrow \infty$) no remanejamento pós-distúrbio. A Figura 3.5 ilustra com diagrama de blocos esta implementação (denominada programa **ranking**).

Essencialmente, este programa compõe-se das seguintes fases:

1. Calcular o ponto inicial através de um fluxo de potência;
2. Resolver o FPO, para um dos dois objetivos disponíveis, para a rede sem alterações (intacta), obtendo-se assim o caso-base $\mathbf{P^0}$;
3. A partir do caso-base, obtido na fase precedente, determinar a classificação da severidade das sobrecargas (obtém-se então o *ranking* das violações ou das sobrecargas);
4. Para cada uma das contingências severas identificadas na fase anterior, executar um FPO com a rede alterada retratando a mudança topológica correspondente. A função objetivo deste PL é o desvio quadrático em relação ao ponto $\mathbf{P^0}$ do caso-base da fase 2; a idéia é remanejar os controles ativos o quanto for necessário para corrigir a violação (ou violações) causada pela contingência em foco, mesmo que para isso seja preciso usar geradores fictícios.

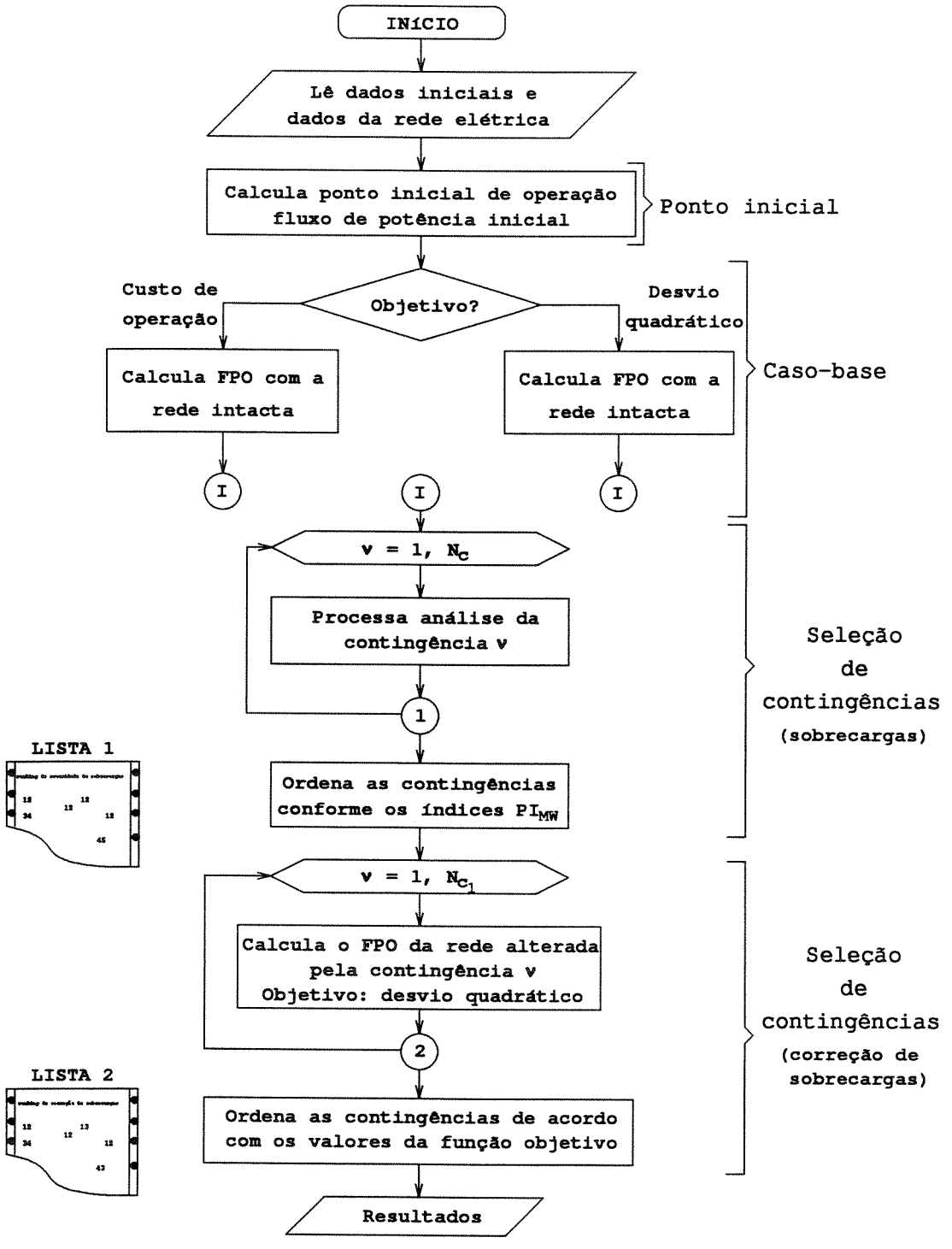


Figura 3.5: Diagrama de blocos do programa ranking.

Ao contrário de uma aplicação de FPORS, que calcula o ponto de operação ótimo-seguro para todas as contingências, o programa **ranking** fornece ao final para o operador um diagnóstico para cada contingência em separado. Neste diagnóstico encontram-se informações a respeito da sobrecarga (PI_{MW}), o valor da função objetivo (designado aqui por PI_{MDQ}; vide equação (2.20)) que indica o mínimo afastamento possível do ponto calculado necessário para corrigir a sobrecarga, o desvio total absoluto em $[MW]$, etc. A Tabela 3.1 mostra um trecho de arquivo de saída deste programa para uma rede do Sistema Interligado Brasileiro.

Tabela 3.1: Trecho de um arquivo de saída emitido pelo programa **ranking**; R significa que não foi preciso usar geração fictícia; rede BR-725 com carregamento leve.

Ordem de severidade		Contingência		Índices		Desvio total	Conceito
sobrecarga	correção	ramo	terminais				
		Nº	NI NF	PI_{MW}	PI_{MDQ}	$[MW]$	R/F
1ª	2ª	808	1119 1167	41,9	0,133	2472,1	R
2ª	4ª	939	1215 1222	29,0	0,044	364,3	R
3ª	3ª	110	182 181	4,3	0,051	49,8	R
4ª	1ª	208	372 830	2,7	0,135	856,2	R

O desvio total absoluto da geração para corrigir a violação decorrente da contingência ν é obtido pela expressão seguinte:

$$\text{Desvio total} = \sum_{j} | P_j^{\nu} - P_j^0 | \quad [MW] \quad \forall \text{ gerador } j$$

Conforme se pode ver no exemplo mostrado na tabela acima, um fato notável neste programa é que ambas as listas críticas obtidas através dos índices de sobrecargas (PI_{MW}) e dos valores da função objetivo de desvio quadrático (PI_{MDQ}) apresentam as mesmas contingências no topo, embora não necessariamente na mesma ordem. Esta constatação é um indicativo de que, o cálculo dos índices PI_{MDQ} através de PLs, realizado pós-seleção de contingências por sobrecargas, também serve para classificar as contingências pelo grau de dificuldade que cada caso apresenta à correção da sobrecarga (i.e., da violação). Portanto, é sugerido neste trabalho a utilização do programa **ranking** como um "filtro" de dois estágios capaz de fornecer, a partir de uma lista "bruta" de contingências, duas outras listas de casos severos, onde são empregados dois critérios: (1) a primeira lista é obtida detectando-se sobrecargas em ramos; e (2) a segunda lista, originária da anterior, é obtida calculando-se o PL da situação pós-contingência com o objetivo de minimizar o desvio quadrático em relação ao caso-base e ao mesmo tempo garantir a viabilidade operacional.

O programa **ranking** apresenta elevado potencial de paralelização na medida em que é essencialmente um *software* de análise de contingências, no qual as tarefas para serem realizadas precisam tão somente do estado básico, da configuração da rede (i.e., matrizes do caso-base) e da lista de contingências. No Capítulo 5, a paralelização deste programa é discutida. Quanto a detalhes de programação paralela (ou distribuída), a descrição dada na seção 4.5.1 é em linhas gerais perfeitamente aplicável ao programa **ranking** paralelo.

Capítulo 4

FLUXO DE POTÊNCIA ÓTIMO COM RESTRIÇÕES DE SEGURANÇA EM UM AMBIENTE DE PROGRAMAÇÃO DISTRIBUÍDA

4.1 Introdução

O planejamento da operação e o controle em tempo-real dos sistemas de energia elétrica exigem computadores com elevada capacidade de processamento. Enquanto as novas metodologias matemáticas ou as técnicas de programação trazem apenas ganhos marginais no desempenho das aplicações, os computadores monoprocessadores modernos, apesar da vertiginosa evolução dos últimos anos, ainda são incapazes de isoladamente satisfazer os rígidos requisitos desta área. Processamento paralelo e distribuído estão entre as novas tecnologias que apresentam grande potencial de aplicação na solução desses problemas.

Processamento paralelo e distribuído são similares, embora sejam concebidos a partir de diferentes conceitos de processamento da informação. Computadores paralelos são geralmente máquinas completas, projetadas especificamente para a finalidade de obter ganhos na velocidade de processamento de tarefas concorrentes. Máquinas paralelas são compostas de múltiplos componentes, como processadores, memórias e discos. Todos estes componentes são integrados num único computador que é usado para executar em paralelo uma aplicação.

Duas categorias principais de computadores paralelos podem ser identificadas: multiprocessador e multicomputador. Um multiprocessador é um computador composto de vários processadores que compartilham uma memória comum. Exemplos destas máquinas são *Encore-Multimax*, *DEC* e *Everex*. Nos computadores de memória compartilhada (multiprocessadores), a comunicação é, em princípio, simples e fácil – cada processador "escreve" sua mensagem num campo da memória global e "conta" aos outros processadores o endereço para que a informação seja encontrada. Um multicomputador é composto de vários processadores cada um com sua própria memória (local e privativa), e se comunicam entre si através de troca de mensagens (i.e., *message passing*). Os multicomputadores são também referidos como máquinas paralelas de memória distribuída. Alguns exemplos dessas máquinas são *nCUBE2*, *SP1* e *iPSC*. Outras formas interessantes de paralelismo são os processadores *pipelined*, usados em computadores vetoriais, especialmente projetados para operações concorrentes de *loops* que aparecem nos cálculos com matrizes e vetores (exemplos: *CRAY YMP* e *IBM 9021/VF*), e os processadores super-escalares que utilizam compiladores especiais para melhorar a execução de programas convencionais [57].

Um sistema de processamento distribuído é uma coleção de computadores autônomos, interconectados por uma rede de comunicação, que trabalham em conjunto para atender as necessidades de processamento de uma companhia, departamento ou laboratório. A Figura 4.1 apresenta uma visão geral de um sistema de computação distribuída.

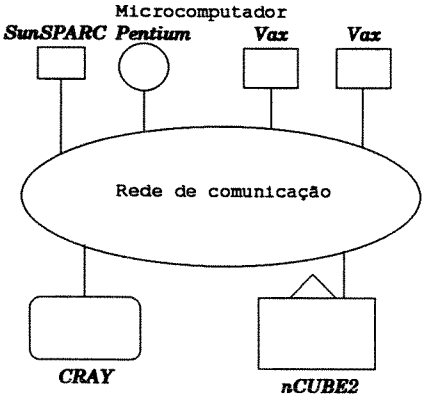


Figura 4.1: Ambiente de computação distribuída.

Este tipo de ambiente computacional descentralizado é viável na atualidade devido à disponibilidade de poderosos computadores-de-mesa (i.e., microcomputadores e estações de trabalho) que podem ser interligados por redes rápidas. Sistemas de processamento distribuído estão substituindo os computadores *mainframe* tanto nas aplicações comerciais como nas científicas.

Os sistemas de computação distribuída podem incluir desde alguns computadores localizados numa sala, conectados por uma rede local (*LAN*) usando cabos coaxiais, ou um grande número de computadores espalhados geograficamente e separados por grandes distâncias (*WAN*), interconectados através de linhas telefônicas, *link* de microondas, cabos de fibras ópticas, etc. Os computadores podem ser de diferentes tipos e capacidades.

Os sistemas de computação distribuída apresentam algumas vantagens sobre os sistemas convencionais. Dentre essas vantagens pode-se destacar: menor custo médio por MIPS em relação a um *mainframe*; os sistemas distribuídos podem ser permanentemente expandidos e permitem a exploração de recursos especiais de *hardware* como equipamentos de alta resolução gráfica ou computadores vetoriais (ou paralelos) de grande capacidade agregada de cálculo por unidade de tempo [57].

Por outro lado, processamento distribuído é um campo novo e repleto de problemas a serem solucionados. Há questões em aberto relacionadas à falta de padronização e segurança. Do ponto de vista das aplicações, existem aspectos associados ao desempenho que se pode atingir com sistemas heterogêneos (i.e., de que forma tais sistemas se comparam às máquinas paralelas dedicadas) e a confiabilidade das implementações em face aos diversos tipos de redes de comunicação disponíveis [31,32,47,60,63,64,68].

Há duas motivações para se pensar na utilização de processamento distribuído em aplicações de sistemas de potência. A primeira é a natureza distribuída (geográfica) do sistema elétrico de potência. A outra motivação é o reconhecimento por parte da indústria de energia elétrica das vantagens de custo e operacionalidade do emprego de microcomputadores e/ou estações, em lugar dos custosos e menos flexíveis computadores *mainframe* [57]. Neste capítulo, o interesse reside na paralelização em um ambiente de programação/processamento distribuído das funções de análise de segurança *on-line*, que são: análise de contingências e o fluxo de potência ótimo.

O fluxo de potência ótimo com restrições de segurança é um típico problema de análise de segurança *on-line*, classificado como um problema de computação intensiva. Conforme foi exposto no Capítulo 2, o FPORS pode ser decomposto em cinco partes: (a) ponto inicial e FPO caso-base; (b) seleção de contingências; (c) ordenação de casos de contingência; (d) otimização; e (e) análise de contingências e geração de restrições. Foi também discutido que o número de contingências pode chegar a ser elevado, o que torna atrativa a resolução do problema de FPORS mediante o processamento distribuído. Observando-se a decomposição apresentada, é possível identificar duas partes do problema onde o processamento distribuído pode ser aplicado sem ter que trabalhar com grãos[1] muito pequenos: (1) seleção de contingências; e (2) análise de contingências e geração de restrições.

O presente capítulo está organizado como segue. Na seção 4.2 é apresentado o sistema *PVM3*, suas componentes e características. Na seção 4.3 são dadas informações sobre medidas de desempenho de aplicações paralelas e distribuídas; são também discutidos aspectos críticos a respeito da avaliação de desempenho. Na seção 4.4 é descrito o ambiente de computação utilizado. A seção 4.5 está dividida do seguinte modo: primeiro, são estabelecidas definições dos paradigmas básicos de programação concorrente; segundo, são descritas as implementações concorrentes dos blocos de seleção de contingências e; por último, é abordado o modelo de programação empregado na paralelização do processo de análise de contingências e geração de restrições para os programas de FPORS.

[1] Granularidade de uma aplicação refere-se à relação $\frac{\text{tempo de execução}}{\text{tempo de comunicação}}$.

4.2 *PVM*: Um Sistema de Programação Distribuída

Ambientes de programação concorrente baseados em redes de computadores frouxamente acoplados podem ser soluções eficazes, viáveis e economicamente atrativas para as exigências computacionais do futuro. Significativos avanços em algoritmos paralelos e arquiteturas têm permitido mostrar o potencial das técnicas de computação concorrente em uma ampla variedade de problemas. A maioria dos esforços, entretanto, tem se concentrado sobretudo em modelos computacionais, versões paralelas de algoritmos, ou arquiteturas de máquinas; somente a partir do final da década de '80 é que alguma atenção foi dada ao desenvolvimento de *softwares* que configuram ambientes computacionais como se fossem verdadeiras máquinas paralelas.

Este aspecto da pesquisa é importante à medida que o processamento paralelo tradicional evolui de soluções isoladas de problemas (matematicamente precisas) para grandes e mais complexos sistemas de *software*. Os requisitos computacionais ideais de uma classe especial aplicações na atualidade são processamento vetorial, processamento paralelo, computador escalar e estações gráficas para visualização de resultados e interação com o usuário [53].

Nota-se que boa parte dos típicos ambientes computacionais já possuem a diversidade de *hardware* requerida para resolver essas grandes aplicações de forma paralela, e também têm suporte para múltiplos modelos de computação concorrente. Redes locais de alta velocidade com estações gráficas, máquinas escalares de alto desempenho, um ocasional multicomputador (um *number crunching*) e talvez um computador vetorial são a norma na atualidade, e não a exceção. Entretanto, para aproveitar esta coleção de recursos são exigidos consideráveis esforços na coordenação entre tarefas e comunicação de dados através de distintos computadores e arquiteturas.

O projeto *PVM* (*Parallel Virtual Machine*) é uma tentativa de fornecer uma estrutura unificada, dentro da qual grandes sistemas paralelos possam ser desenvolvidos de uma maneira fácil e eficiente. *PVM* é um sistema de *software* que permite que uma rede seja usada como um único recurso computacional concorrente. Assim, grandes problemas podem ser resolvidos utilizando-se a força computacional agregada de várias máquinas.

O desenvolvimento do *PVM* iniciou-se no verão de 1989, no Oak Ridge National Laboratory (ORNL), e prossegue sendo aperfeiçoado por meio de esforços conjuntos que envolvem diversos pesquisadores [52].

Na mesma linha do *PVM*, esforços independentes têm produzido outros sistemas de *software* para programação distribuída. Alguns exemplos são: *MPI*, *Linda*, *HBD*, *VHAM*, *P4*, *POSYBOL*, *TCGMSG*, *Amber*, *Paralation Lisp*, *Nectar*, *VISTAnet*, etc. [68].

Neste trabalho, a versão utilizada do sistema *PVM* é a 3 e suas subdivisões (3.3.5, $\cdots$, 3.3.11). Doravante, no texto, é usada a designação *PVM3* sempre que se referir a aspectos específicos do sistema.

4.2.1 Componentes e Principais Características

O sistema *PVM3* é composto de duas partes. A primeira parte é um processo *daemon*, chamado **pvmd3** ou **pvmd**, que "reside" em todos os computadores que formam a máquina virtual. Este *daemon* foi projetado de modo que qualquer usuário com *login* pode instalá-lo em uma máquina. Quando um usuário deseja processar uma aplicação, ele executa **pvmd3** em um dos computadores, que, em seguida, inicializa **pvmd3** nas demais máquinas que compõem a máquina virtual (definida pelo usuário). O computador onde o usuário executa em primeiro lugar o *daemon* do *PVM3* é designado neste trabalho por *host-login* (ou nó-0). A segunda parte do sistema é uma biblioteca de rotinas *PVM3* de interface (**libpvm3.a**). Esta biblioteca contém rotinas que são chamadas na aplicação do usuário como simples instruções de programação. Estas rotinas permitem ao programador da aplicação distribuída (paralela) fazer a transferência de mensagens, alocar (ou "desovar") processos em processadores, coordenar tarefas, modificar a máquina virtual, prevenir falhas de computadores ou de processos, etc. Portanto, a aplicação que possui instruções do *PVM3* deve ser compilada ligando a biblioteca **libpvm3.a**. A Figura 4.2 mostra uma rede local de estações de trabalho de alta *performance* transformada numa máquina paralela provida com o sistema *PVM*.

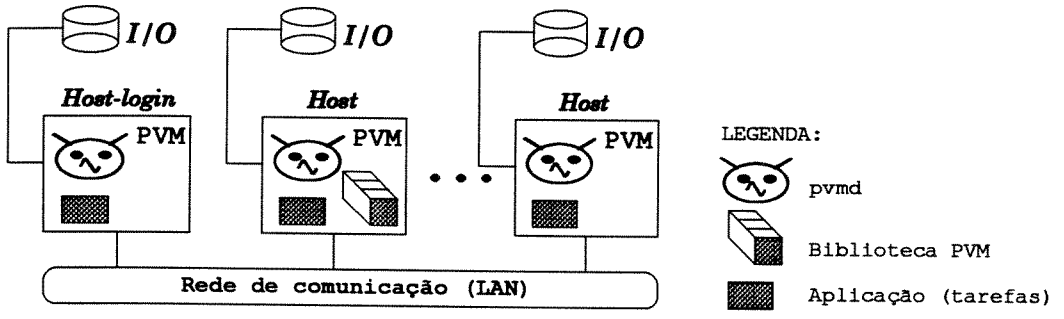


Figura 4.2: Máquina paralela virtual com o sistema de *software PVM3*.

A aplicação *PVM3*, para ser executada, pode ser inicializada do *prompt* do *UNIX* de qualquer um dos computadores configurados. Múltiplos usuários podem configurar máquinas virtuais, inclusive com sobreposição de *hosts*.

Algoritmos concorrentes de aplicações são expressos usando o paradigma mais conveniente e o sistema *PVM3* executará esses programas nos computadores mais apropriados disponíveis. Se o desejo for incorporar uma aplicação existente a um grande sistema de aplicações com pouca ou nenhuma modificação, o *PVM3* é projetado para possibilitar isto de um modo conveniente e natural.

O sistema *PVM3*, por meio de sua biblioteca, fornece um conjunto de primitivas de interface com o usuário que podem ser incorporadas dentro das linguagens procedurais FORTRAN e C. Essas primitivas existem para evocar processos, enviar/receber dados, sincronizar via barreiras e

compartilhar memória. Processos podem ser iniciados no modo síncrono ou assíncrono, e serem condicionados ao início ou término de outro processo, ou sobretudo à disponibilidade de dados. A transmissão de mensagem e a saída em arquivo podem ser precedidas por chamadas especiais que asseguram que o dado é transmitido ou armazenado de maneira independente do tipo da máquina. Portanto, as construções do *PVM3* permitem que os mais apropriados paradigmas de programação e linguagens sejam usados para cada componente de um sistema paralelo, enquanto preservam a habilidade dos componentes de interagir e cooperar.

A rede para o *PVM3* pode ser local ou do tipo *WAN*, ou uma combinação de ambas, e o *pool* de *hosts* pode ser variado de forma dinâmica. Os *hosts* podem ser máquinas escalares, estações, ou computadores paralelos. Os requisitos gerados pelas construções ao nível do usuário e as ações necessárias são interpretadas pelo *PVM3* de uma maneira independente da máquina. O *PVM3* consiste essencialmente de uma coleção de algoritmos de protocolos, cujas funções são implementar confiável e seqüenciada transferência de dados, exclusão mútua, etc. Na tentativa de fazer o sistema tão robusto quanto possível, esses algoritmos também incorporam mecanismos de detecção de erros (notificação de falha de processo e processador é fornecida para as aplicações), conseqüentemente, habilitando-o a tomar ações corretivas. Esta característica de detecção de erros foi utilizada neste trabalho em associação com as propriedades dinâmicas do *PVM3*; a implementação está descrita no Capítulo 5.

Em síntese, as principais características do *PVM3* são:

i) interface com o usuário atualizada e de alto nível;

ii) identificador inteiro de tarefa paralela;

iii) permite a alocação e controle de processos;

iv) permite configurar e re-configurar a máquina de dentro da aplicação em tempo de execução;

v) tolerância a falha pode ser implementada;

vi) permite a programação de aplicações como grupos dinâmicos;

vii) tarefas *PVM3* podem receber sinais ou notificações de outras tarefas ou do sistema *UNIX*;

viii) a transmissão se faz através do envio direto com endereço fixo ou através de difusão de dados, e a recepção pode ser do tipo bloqueante ou não-bloqueante, o que permite a programação no modo assíncrono;

ix) pode ser integrado com sistemas paralelos dedicados, e suas primitivas são muito parecidas com as primitivas de diversas máquinas paralelas, o que facilita a portabilidade de aplicações de diferentes ambientes [32].

Uma característica interessante do *PVM3* é que ele não requer que as estações sejam dedicadas à aplicação e, assim, a máquina paralela virtual pode trabalhar concorrentemente com outras aplicações (de outros usuários ou do próprio usuário). É também possível executar vários processos de um único programa distribuído no mesmo processador [63].

4.3 Medidas de Desempenho

Para a avaliação do desempenho de sistemas paralelos ou distribuídos (i.e., programas, computadores, algoritmos paralelos, etc.), as seguintes medidas são particularmente importantes: o *speedup*, a eficiência e a eficácia [64].

Seja p o número de processadores (*hosts* ou nós) e W o tamanho ou dimensão do problema[2]; o *speedup* S é a medida do ganho (em tempo) do programa paralelo executado com p processadores relativamente ao melhor programa seqüencial, ambos na mesma aplicação. A definição do *speedup*, seguindo uma prática comum das publicações, é modificada: toma-se o *speedup* em relação ao tempo medido do próprio programa paralelo em um processador ($T(1)$). Assim, a fórmula do *speedup* é dada pela equação (4.1).

$$S \triangleq \frac{T(1)}{T(p)} \tag{4.1}$$

Onde, $T(p)$ é o tempo global de execução do programa paralelo usando p processadores.

Em muitas publicações da área é costume referir ao tempo medido em programas paralelos como tempo de UCP (*CPU time*) e, em outras, como *clock time*, *user time*, *real time*, etc., e não são especificadas as componentes dessas medidas. Os *speedups* mostrados geralmente aproximam-se do ideal, isto é, do número de processadores p. Neste trabalho, toma-se o cuidado de definir o tempo medido nas aplicações paralelas como o tempo físico (isto é, o mesmo tempo de relógio) efetivamente transcorrido do início da execução do programa até a sua conclusão. Obviamente, o valor dessa medida dependerá, além da aplicação, das condições em que são feitos os testes: se diretamente do *prompt* do *UNIX*, se de dentro de uma "janela" do *X-Windows*, ou em *background*. Particularmente em sistemas de processamento distribuído é importante saber se o processador em uso está sendo compartilhado com processos de outros usuários. Neste caso, pode-se identificar duas condições: rede livre ou rede ocupada. A condição ocupada logicamente compreende vários níveis de utilização difíceis de quantificar. Entretanto, acredita-se que o uso compartilhado não é de interesse para aplicações práticas *on-line* do processamento distribuído em Centros de Controle. Talvez, o uso compartilhado possa ser útil para o modo estudo em aplicações de planejamento, ou outras aplicações intensivas, onde o objetivo seja aproveitar recursos computacionais em horários de baixa utilização (e.g., finais de semana e feriados).

Diante do exposto, em geral, o tempo global de execução do programa paralelo ($T(p)$), por exemplo, medido no processo do *host-login* (ou do nó−0), é composto de três parcelas, conforme indica a equação (4.2) [46].

$$T(p) = T_{I/O} + T_{\text{proc}}(p) + T_{\text{com}}(p) \tag{4.2}$$

Onde,

[2] W pode ser entendido como o tempo total gasto para processar do início ao fim, de forma seqüencial, um dado programa em um determinado computador.

$T_{I/O}$ é o tempo para ler e escrever em arquivos, normalmente feito apenas pelo *host-login* ou nó–0;

$T_{\text{proc}}(p)$ é o tempo gasto efetivamente com cálculos numéricos, operações de decisão, *loops* e acessos a posições de vetores na memória;

$T_{\text{com}}(p)$ é o tempo de envio/recepção de mensagens.

Usualmente, $T_{\text{proc}}(p)$ é uma função que decresce com o aumento de p, enquanto que $T_{\text{com}}(p)$ aumenta, e $T_{I/O}$ depende do tamanho do problema (notadamente, em aplicações de redes elétricas). No Capítulo 6, de testes de resultados, os tempos tomados para medidas de desempenho são especificados.

A eficiência E é uma medida da fração do tempo em que os recursos estão ocupados durante a execução paralela da aplicação. Matematicamente, E é a relação entre o *speedup* e o número de processadores.

$$E \triangleq \frac{S}{p} = \frac{T(1)}{pT(p)} \tag{4.3}$$

A eficácia η descreve o grau de utilização dos processadores e é definida pela equação (4.4).

$$\eta \triangleq \frac{S^2}{p} \tag{4.4}$$

À medida que p cresce, a eficácia aumenta até um valor máximo e depois decresce. O número de processadores que corresponde ao η máximo é designado como *pws* (*processor working set*), isto é, o número de processadores para o qual a relação entre o benefício (aumento no *speedup*) e o custo (decréscimo da eficiência) é máxima.

A eficiência seria igual a 1 se o *overhead* de explorar o paralelismo (leitura de dados, comunicação e chamadas adicionais de primitivas) fosse negligenciado e o trabalho fosse distribuído igualmente entre as UCPs, implicando conseqüentemente que nenhum processador ficaria ocioso. Esta situação evidentemente é ideal e, portanto, inatingível. Um objetivo mais realista deve almejar uma eficiência, quanto possível, longe de zero, conforme o crescimento de p e W. Assim, pode-se concluir que a eficiência E dá uma medida do grau de aproximação ao ganho ideal.

Uma questão levantada há algum tempo é, dado um problema a ser paralelizado cuja fração essencialmente seqüencial é f, qual o máximo *speedup* atingível se o número de processadores p for feito arbitrariamente grande. Desta questão decorre a Lei de Amdahl, que pode ser quantificada da seguinte forma: se o programa possui duas partes, uma que é inerentemente seqüencial e

outra que é totalmente paralelizável, e se a parte seqüencial consome a fração f do tempo de computação total, então o *speedup* é limitado como indica a expressão (4.5).

$$S \leq \frac{1}{f + \frac{(1-f)}{p}} \leq \frac{1}{f} \tag{4.5}$$

A expressão (4.5) estabelece que o máximo ganho do programa paralelo estará limitado pela fração não paralelizável do código.

Existe uma considerável discussão sobre o valor do máximo *speedup* que se consegue alcançar em situações reais e da validade da Lei de Amdahl [46,64]. Esta lei parece, em princípio, desencorajar o uso de soluções paralelas (e distribuídas), especialmente os sistemas massivamente paralelos. Entretanto, a prática tem mostrado que não é esta toda a verdade sobre o assunto.

Outro aspecto interessante é o uso do processamento concorrente na abordagem de problemas de otimização combinatorial, cujas soluções paralelas através de heurísticas tendem a ser mais eficientes que o esperado. Além disso, nos anos recentes, surgiram as técnicas de multi-agentes que se baseiam numa mixagem de diferentes métodos para resolver simultaneamente um mesmo problema, geralmente do tipo *NP-hard*. A simbiose desses métodos, quando bem programados em ambientes de computação paralela ou distribuída, resulta geralmente em surpreendentes efeitos sinérgicos (relativo a sinergia). São os chamados times assíncronos (*A-Teams*) [38,50].

O desempenho de algoritmos paralelos é limitado pelos atrasos introduzidos pelo próprio programador da aplicação ou decorrentes das características dos sistemas de computação. Isto é conhecido pelo nome genérico *overhead*. Há, fundamentalmente, três tipos de *overhead*: (1) de sincronismo; (2) de comunicação; e (3) de algoritmo. O primeiro é originado pelo desbalanceamento de carga entre os vários processadores do sistema, que impõem tempos de espera indevidos durante a execução. Geralmente, este *overhead* advém de uma decomposição ruim do problema. O *overhead* de comunicação decorre da limitada capacidade de comunicação do sistema paralelo e da latência inerente ao início do transporte da mensagem [48]. Este inconveniente pode ser minimizado pela escolha adequada do tamanho das tarefas observando-se a capacidade de comunicação do *hardware* (i.e., granularidade). O algoritmo paralelo pode ter mais ou menos operações que o algoritmo seqüencial para resolver o mesmo problema. A habilidade do programador e o conhecimento do problema são os melhores instrumentos para se reduzir o *overhead* de algoritmo. Outros *overheads* são a localização dos dados e a forma de armazenamento [56].

Estabelecida a noção de *overhead*, uma expressão bastante realista e útil na previsão de ganhos da paralelização de programas pode ser introduzida. Seja T_O o tempo total do *overhead* paralelo, isto é, a soma de todos os atrasos que se originam da execução paralela do algoritmo; neste tempo estão incluídos custos de comunicação, operações não essenciais e tempos ociosos devido à sincronização e às componentes seqüenciais do algoritmo. Para um dado sistema paralelo, T_O é usualmente uma função do tamanho do problema (W) e do número de processadores

(p), portanto, pode ser escrito como $T_O(W,p)$. Assim, $T_O(W,p) = pT(p) - W$. É fácil verificar que o *speedup* é dado pela expressão (4.6).

$$S = \frac{p}{(1 + \frac{T_O}{W})} \tag{4.6}$$

A equação (4.6), ao ser analisada, fornece muita informação. Mostra que o *overhead* aparece como um redutor aplicado sobre o *speedup* ideal e, quanto maior for W com T_O fixo ou mantido em níveis baixos, melhores serão os ganhos. Isto é um fato. Muitas publicações mostram resultados de programas paralelos, cujas tarefas são quase totalmente independentes, testados em grandes problemas ou bancos de dados enormes, que exibem formidáveis *speedups*. O mesmo programa paralelo, no mesmo número de processadores, executando problemas pequenos exibirá ganhos modestos ou até nenhum ganho [37].

4.4 Ambiente Computacional Utilizado

O trabalho de construção de rotinas, depuração de algoritmos, testes de eficiência, testes das primitivas de programação concorrente, confecção dos programas seqüenciais e posterior transferência para ambientes de computação distribuída foi inteiramente desenvolvido no Laboratório de Sistemas de Energia Elétrica (LSEE), do Departamento de Sistemas de Energia Elétrica (DSEE), da Faculdade de Engenharia Elétrica e de Computação da UNICAMP.

O ambiente de computação disponível neste laboratório compreende os seguintes computadores:

- Estações de trabalho interligadas pelo meio *Ethernet*, operando com uma versão *SunOS* do *UNIX*, perfazendo o total de 12 máquinas; sendo uma *SunSPARC10*, uma *SunSPARC Server330*, e as demais *SunSPARC2*. Nesta rede, ainda se encontram conectadas duas outras máquinas *SunSPARC ULTRA*, operando com uma versão *Solaris* do *UNIX*.
- Estações *SunSPARC4* e *SunSPARC20* conectadas por cabos de fibras ópticas (*FDDI*) e operando sob o *Solaris*. Sendo 8 máquinas *SPARC4* e uma *SPARC20*.
- Computador paralelo *nCUBE2*, de 64 processadores, ligado à rede local através da estação *SunSPARC Server330*, cujo acesso é permitido a partir de qualquer máquina da rede.

Todos estes sistemas de computadores estão interligados entre si e à rede da faculdade. A Figura 4.3 fornece uma visão geral deste parque computacional.

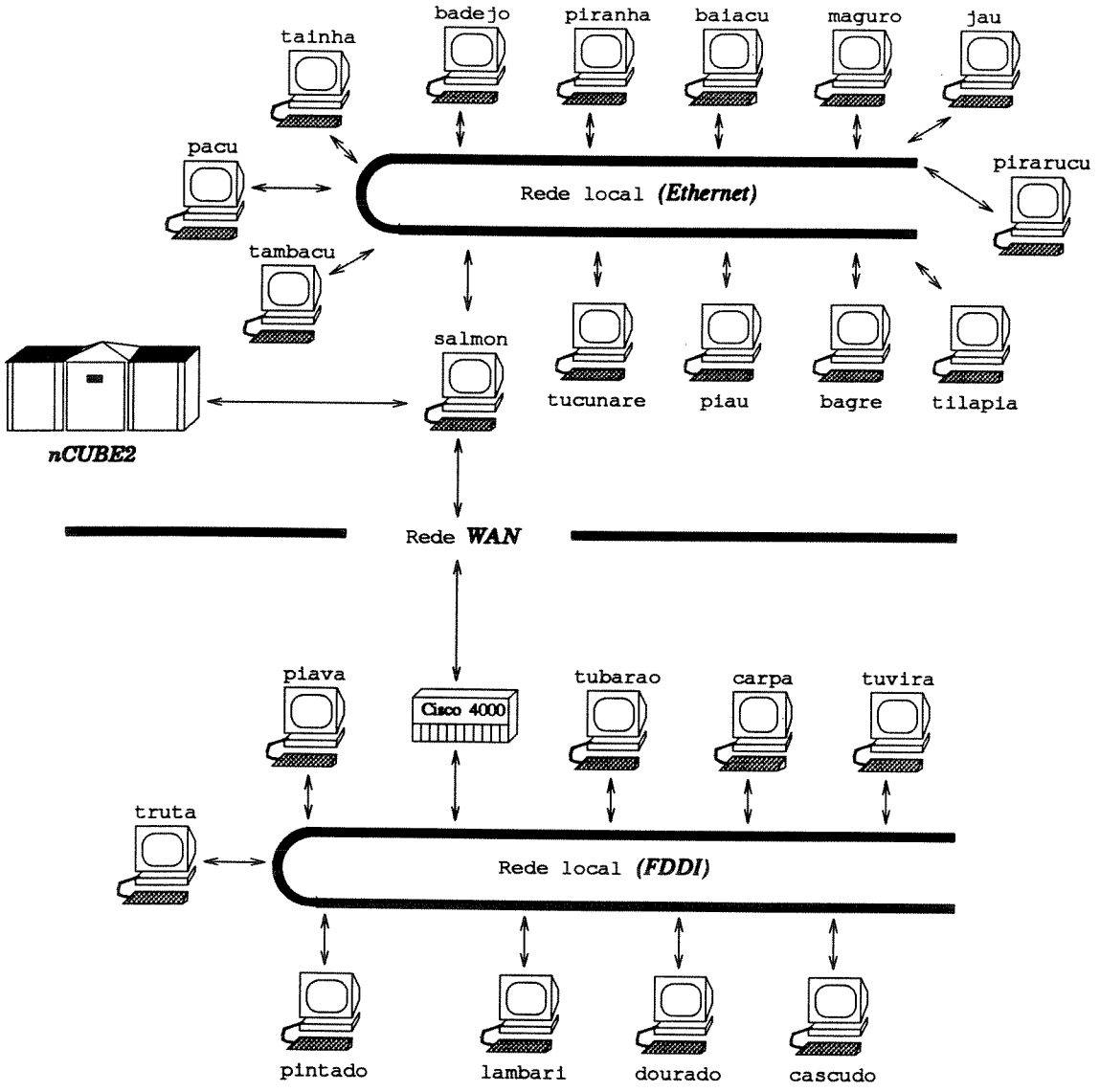


Figura 4.3: Ambiente computacional do LSEE/UNICAMP.

Outro sistema no qual se teve acesso durante o desenvolvimento deste trabalho é o conjunto de máquinas de última geração do Centro Nacional de Processamento de Alto Desempenho em São Paulo, CENAPAD-SP, instalado nas dependências da universidade. Este sistema anexo é composto essencialmente de estações *IBM RISC/6000* e de nós do computador paralelo da *IBM*, o *SP1*, recentemente atualizado para *SP2*. O sistema do CENAPAD-SP em quase sua totalidade é configurável para processamento distribuído com o *PVM3* e com o *MPI*.

Na fase de testes das implementações todos estes recursos foram de alguma forma utilizados, seja para fazer comparações de desempenho, verificar a portabilidade de programas, ou para testar os próprios algoritmos paralelos funcionando numa rede heterogênea com grande quantidade de processadores.

4.5 Implementações Concorrentes do Fluxo de Potência Ótimo com Restrições de Segurança

As implementações concorrentes dos programas apresentados no Capítulo 2 no ambiente de processamento distribuído, criado com o *PVM3* e uma rede de estações, são descritas nesta seção.

Quaisquer dos três programas, FPORS-I, FPORS-II e FPORS-III têm suas implementações distribuídas elaboradas segundo uma filosofia comum:

- O bloco de seleção de contingências, inerentemente paralelo, foi decomposto em tarefas de granularidade alta, onde cada *host* processa uma contingência por vez, com as opções do modo síncrono e do modo assíncrono. A seção 4.5.1 descreve os principais aspectos desta fase dos programas paralelos (i.e., distribuídos).
- Logo após o bloco de seleção de contingências, segue o bloco composto de otimização e geração de restrições precedida de análise de contingências (vide fluxogramas das Figuras 2.6, 2.7 e 2.9). Esta fase dos programas tem algum grau de paralelismo que pode ser explorado, especialmente o nível superior de relaxação, conforme descrito anteriormente na seção 2.4.2. Portanto, em todos os programas, o processo de geração de restrições foi paralelizado. A seção 4.5.2 descreve a paralelização desta fase dos três programas de FPORS.

O processo de análise de contingências tem boa granularidade e apresenta independência entre tarefas. Cada caso de contingência a ser processado equivale ao cálculo de um fluxo de potência (com salvaguardas de compensação, etc.) e necessita basicamente do estado atual do sistema (vetor de estado).

Durante o processo de otimização no FPORS, é sabido que o número de restrições de segurança que ficam ativas (i.e., na base do PL) no ponto ótimo é relativamente pequeno. Este fato sugere que, durante as iterações do FPORS (passos **c)** e **d)** descritos na seção 2.3), a quantidade de vezes que o estado do sistema deve ser remetido ao processo de análise de contingências e geração de restrições é também pequena. A independência do tratamento dos cenários pós-contingências não é completa, uma vez que a solução do caso-base (ou problema de investimento) é alterada como resultado da adição de novas restrições, à medida que o processamento evolui para a solução final. Isto significa que, pode acontecer que as contingências tenham que ser novamente verificadas para o novo ponto de operação. Felizmente, esta dependência é muita fraca, porque o ponto de operação é modificado um número de vezes relativamente menor que o total de contingências a serem analisadas.

Outro aspecto interessante, e que favorece o modelo de paralelização aqui proposto, é a característica "dependente" observada nas contingências mais freqüentes associadas ao subproblema ativo. Ou seja, é comum que a eliminação de uma violação, especialmente a maior, implique na eliminação de outras violações.

Modelos de Programação Concorrente

A utilização da computação paralela requer modelos apropriados de programação (também chamados de paradigmas). O modelo de programação é uma visão abstrata de como os computadores paralelos são programados. Em virtude dessa abstração ser independente do ambiente físico onde o programa será executado, ela é muito útil para o desenvolvimento do algoritmo concorrente; neste tipo de apresentação os detalhes do *hardware* ficam escondidos, o que auxilia a portabilidade da aplicação.

Dois paradigmas de programação são comuns nos estudos de processamento paralelo ou distribuído: um modelo centralizado chamado *Master/Slave*; e um modelo descentralizado baseado no modelo *SPMD, Single Program Multiple Data.*

Modelo *Master/Slave*:

O modelo *Master/Slave* é composto de quatro elementos básicos: a tarefa mestre (geralmente residente no nó-0 ou no *host-login*, processo P_0), as tarefas escravas (P_1, P_2, $\cdots$ P_i, $\cdots$, P_{p-1}), comunicação e base de dados. Tradicionalmente, o programa mestre tem característica "burocrática", isto é, quase não executa operações de cálculo e tem a função principal de coordenar os demais processos. Esta ociosidade do processo P_0 deve ser combatida uma vez que ela repercute em baixa eficiência do modelo.

Modelo *SPMD*:

O modelo *SPMD* é composto de processos idênticos, comunicação e base de dados. Um exemplo típico, que ilustra bem esta forma de programação concorrente, é o cálculo em paralelo

de um somatório de números. Neste exemplo, cada processo atua sobre a porção que lhe cabe no cômputo da soma global, normalmente executando igual quantidade de operações aritméticas. A comunicação se faz no início do processamento (i.e., *fork*) e no fechamento das operações (i.e., *join*).

Estes são os paradigmas básicos de qualquer sistema paralelo ou distribuído e, embora não sejam adequados a todas as categorias de aplicações, constituem o ponto de partida de incontáveis programas concorrentes. Observa-se, também, da literatura de programação paralela e de suas aplicações que não há padrão ou rigor quanto ao uso e construção dos modelos de programação. Dado um algoritmo seqüencial com potencial de paralelização, procura-se o melhor mapeamento desse algoritmo no ambiente paralelo disponível, seja uma combinação proveitosa dos modelos *Master/Slave* e *SPMD* ou algum outro modelo mais complexo. Para esclarecer melhor o assunto, o leitor é convidado a ler o próximo capítulo.

Nas implementações de programação distribuída do FPORS foi utilizada uma mistura desses conceitos. Conforme exposto anteriormente, na fase de obtenção do *ranking* das contingências foi empregado um híbrido dos modelos descritos acima. Já na fase do cálculo do FPORS propriamente dito, o processo residente no *host-login* (processo P_0) é caracterizado como Consumidor, porque gera novos estados de operação da rede à medida que consome restrições de segurança oriundos dos processos remotos. Ainda nesta fase do FPORS, os processos P_1, P_2, $\cdots$ P_i, $\cdots$, P_{p-1} são designados por Produtores. São assim chamados porque ao receberem novo estado calculam contingências das suas listas privativas e produzem restrições para envio ao processo Consumidor (P_0). Dessa forma, esta fase do FPORS é resolvida no modo assíncrono segundo o modelo Produtor-Consumidor.

Cada implementação paralela (distribuída) do FPORS consiste de um único código que processa a fase de obtenção do *ranking*, isto é, todas as etapas preliminares descritas na Figura 2.5, e os cálculos subseqüentes que envolvem as subtarefas de otimização, análise de contingências e geração de restrições. Todas as implementações concorrentes deste trabalho foram programadas segundo a modalidade grupos dinâmicos. Esta modalidade, oferecida pelo *PVM3* nas versões mais recentes, possibilita o emprego de rotinas de configuração dinâmica e de facilidades na comunicação dos dados, como PVMFBCAST(). A Figura 4.4 ilustra o aspecto geral dos programas *PVM3* elaborados. Nesta figura, o bloco designado por "FAZ CÁLCULOS INTENSIVOS EM PARALELO" corresponde ao processamento paralelo das tarefas de seleção/análise de contingências e cálculo do FPORS propriamente dito. Os componentes e a maneira como é programado este bloco estão descritos nas seções 4.5.1 e 4.5.2.

Na Figura 4.4, os símbolos **e** e **r** significam, respectivamente, envio e recebimento. O Apêndice C relaciona as principais primitivas do *PVM3* e fornece os seus significados.

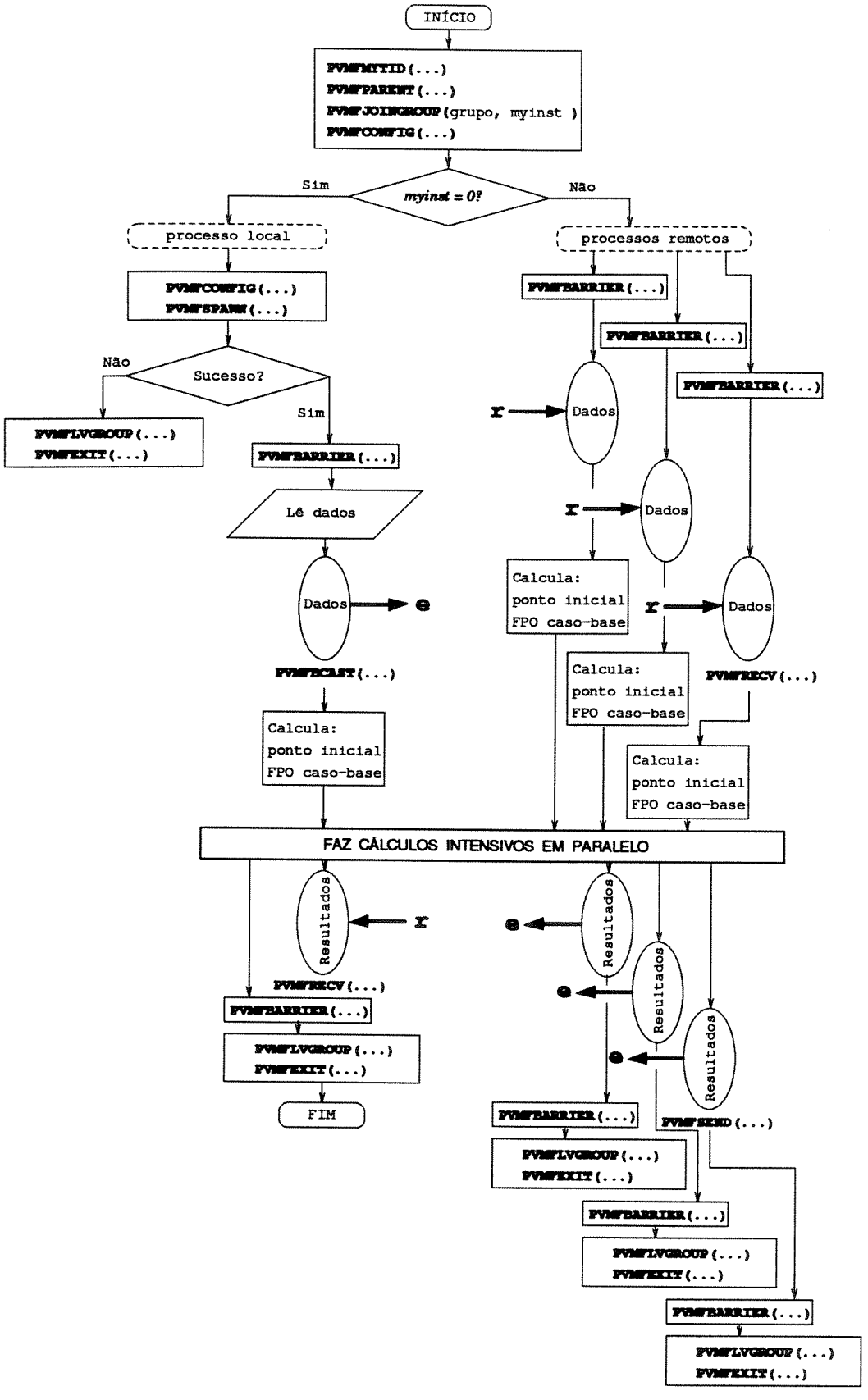


Figura 4.4: Fluxograma geral dos programas concorrentes (1 processo local e 3 remotos).

4.5.1 Programação Distribuída da Análise de Contingências

Nesta aplicação utiliza-se um modelo de programação híbrido: *SPMD* e *Master/Slave*. O modelo híbrido é um modelo *SPMD* no sentido clássico de que, para a parte de computação intensiva do código, há um único programa e múltiplos dados. Entretanto, existem algumas tarefas que executam somente em um nó especificado (estação *host-login* ou nó−0) o que confere um aspecto *Master/Slave* ao modelo. Isto é assim porque o nó responsável pela tarefa *master* também processa tarefas *slave* sempre que for possível. A principal vantagem desta técnica é garantir boa eficiência até mesmo para reduzido número de processadores, o qual nem sempre acontece quando se utiliza o modelo centralizado *Master/Slave* ("puro") da forma como é comumente conhecido [55].

O pseudo-código da Figura 4.5 ilustra como este modelo híbrido é implementado em um único programa.

```
c   Inicializa Processos;
c   Prepara Dados e Calcula FPO Caso-base;
        ...
        ...
        IF( myinst = 0 ) THEN

            call MASTER()        ! coordena e processa casos

          ELSE IF( myinst ≠ 0 ) THEN

            call SLAVE()         ! processa casos e envia resultados

        END IF

c   Finaliza Processos ou Continua
```

Figura 4.5: Pseudo-código da parte intensiva do programa distribuído (modelo híbrido para análise de contingências).

No que se refere à comunicação, os programas de análise de contingências oferecem duas opções: (1) modo síncrono; e (2) modo assíncrono.

Os cálculos da análise de contingências são, por natureza, computacionalmente desbalanceados. Numa lista típica há casos de contingências simples e casos de múltiplas, e também contingências que levam a ilhamentos do sistema. Se a solução pós-contingência completa (AC) for requerida a partir de casos pré-selecionados (por exemplo, após o *screening* de contingências), com muito mais razão é que se tem o desbalanceamento nos cálculos. Então, um modelo de comunicação assíncrono deve ser considerado, motivado pelo fato de que a solução desses casos pode exigir uma ampla faixa de tempos de execução, resultando no desbalanceamento da carga. As contingências mais críticas podem gastar mais iterações que as menos críticas.

No modo síncrono é pressuposto que os casos de contingências exigem similares tempos de execução. Ora, sabe-se do exposto acima que isto nem sempre é verificado. Contudo, se os tempos diferirem muito pouco entre si pode ser compensador utilizar o modo síncrono. Nesta modalidade, os casos de contingências da lista são previamente repartidos, eqüitativamente, entre os *hosts*. Por exemplo, se estão disponíveis os *hosts* simbolizados por □, △ e ○ para processar dez casos, 1, 2, 3, ···, 10, faz-se a divisão de tarefas como ilustrada na Figura 4.6.

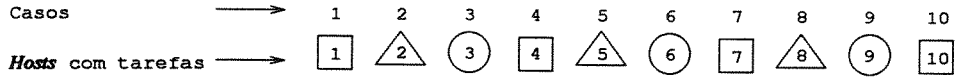


Figura 4.6: Distribuição de tarefas entre *hosts*, modo síncrono.

Observa-se da Figura 4.6 que nenhum caso de contingência ficou sem ser processado, apesar da divisão não ser exata.

A distribuição de tarefas indicada na Figura 4.6 é conseguida com a programação trivial ilustrada na Figura 4.7.

```
...
DO k = myinst + 1, Nc, NPROC
   ncaso = lista(k)          ! toma um caso da lista

   call CONTING()            ! calcula o caso ncaso
   call STORE()              ! guarda resultado para enviar depois
END DO

...
```

Figura 4.7: Programação do modo síncrono para calcular N_c contingências em $NPROC$ processos distribuídos ($myinst = 0, 1, 2, \cdots, NPROC - 1$).

No processamento síncrono, os *hosts* ficam de posse da lista global de contingências. Para cada caso processado, o *host* armazena localmente em vetores os resultados obtidos (por exemplo, índices de severidade). Ao final da análise, os *hosts* enviam os resultados ao *host-login*, que os organiza e adiciona aos seus próprios resultados.

O modo assíncrono trata as contingências como um *pool* de tarefas. O processo do *host-login* coordena toda execução, distribui tarefas, recebe *flags* de casos/*hosts* concluídos, autoriza a parada e ainda processa casos. No início, todos os *hosts* são alimentados, cada qual com um caso de contingência. Assim que um *host* termina sua tarefa, ele requisita uma nova. Neste momento, o processo do *host-login* pode estar calculando contingência, verificando o *buffer* ou atendendo a solicitação de outro *host*. Os fluxogramas mostrados na Figura 4.8 ilustram a programação deste paradigma.

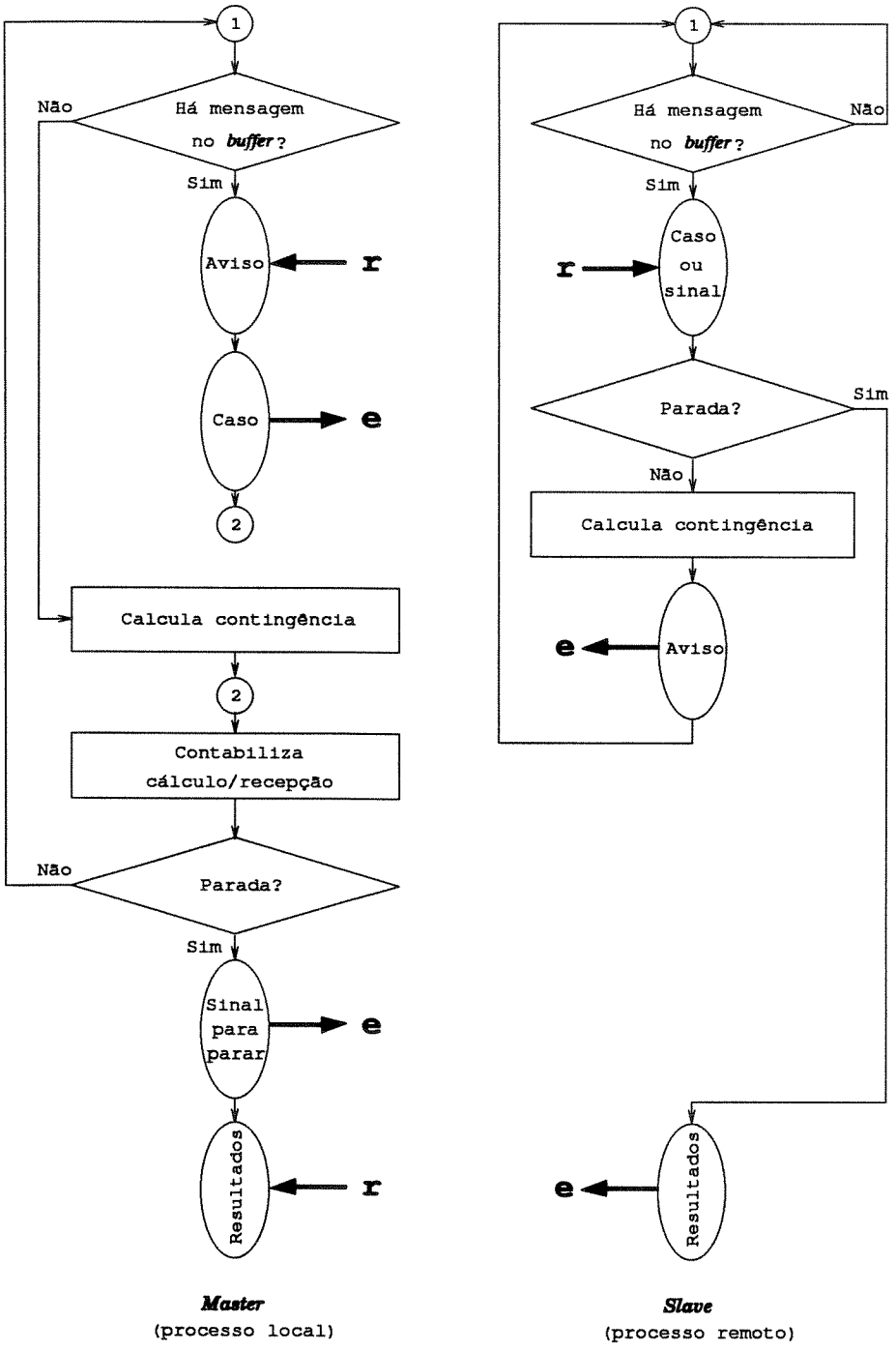


Figura 4.8: Diagrama de blocos das rotinas *master* e *slave* no modo assíncrono.

A vantagem do modo assíncrono está na redução do tempo de espera dos processadores, se tarefas desbalanceadas forem calculadas. Já no modo síncrono, um processo, assim que termina suas tarefas, tem que aguardar a conclusão dos demais, ou seja, há uma espécie de ociosidade dos processadores. Todavia, programas assíncronos requerem elevado grau de coordenação, que repercute no aumento da comunicação. Se haverá ganho ou não em relação ao modo síncrono isto vai depender do quão acentuado é o desbalanço das cargas computacionais das tarefas, e também das características de comunicação do ambiente de processamento distribuído. A Figura 4.9 mostra dois diagramas de execução para três processadores e seis tarefas, onde são comparados os dois modos de comunicação.

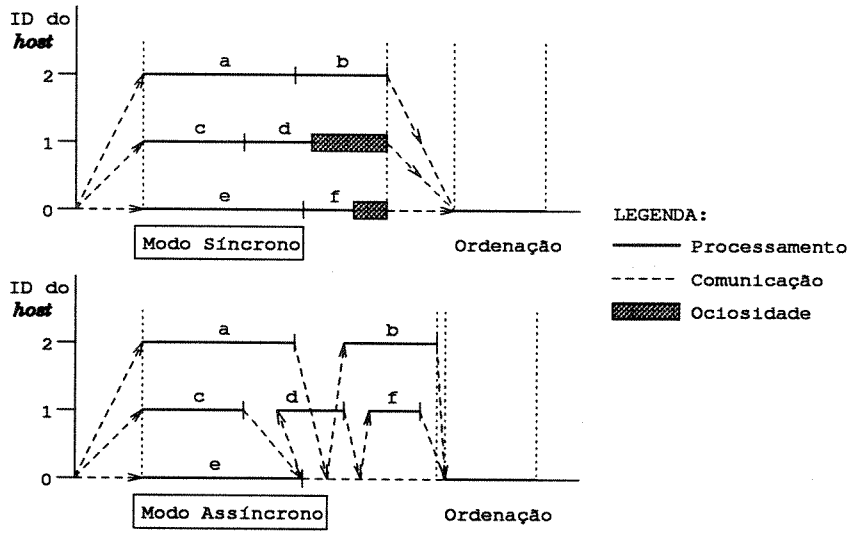


Figura 4.9: Comparação dos perfis de execução dos modos síncrono e assíncrono para aplicações concorrentes de análise de contingências.

Uma variante do modo assíncrono foi também implementada, que permite o envio de mais de um caso por vez ao *host* solicitante. Em outras palavras, pode-se enviar um pacote de tarefas (i.e., "envelope") ao invés de uma só. A idéia por trás dessa implementação consiste em reduzir a freqüência com que se solicita a rede de comunicação durante o processamento das contingências. Isto é particularmente importante em sistemas paralelos não dedicados, como o *PVM3* operando em uma rede de computadores.

Para facilitar a referência no texto, a modalidade de envio de uma tarefa por vez é chamada de modo assíncrono 1, e o envio de um pacote (de mais de uma tarefa) é designada por modo assíncrono 2. Comparações entre os modos síncrono e assíncrono quanto ao desempenho são estudadas no Capítulo 6.

4.5.2 Modelo Produtor-Consumidor no Processamento Iterativo do FPORS para as Contingências Críticas

Esta seção descreve a paralelização das operações do FPORS representadas nos diagramas de blocos ilustrados nas Figuras 2.6, 2.7 e 2.9.

No processamento distribuído dos programas de FPORS, uma adequada decomposição do problema em subtarefas para serem executadas por diferentes *hosts* é fundamental para se obter bom desempenho. A maneira como a decomposição é feita varia com o programa em foco. Em linhas gerais, a decomposição proposta é essencialmente a mesma estudada em [43,56]. O modelo Produtor-Consumidor[3] apresenta nestas implementações a seguinte partição de tarefas:

Tarefa Consumidora de Restrições (Processo do *Host-login* ou Local):

Subtarefas:

- Inicialização das tarefas produtoras;
- Algoritmo de otimização;
- Coordenação das tarefas e decisão de parada mediante um critério.

Tarefa Produtora de Restrições (Processo dos Demais *Hosts* ou Processos Remotos):

Subtarefas:

- Análise de contingências;
- Geração de restrições.

As subtarefas de otimização e geração de restrições são executadas no modo assíncrono, em um ou mais processadores. Cada uma dessas subtarefas produz dados que são consumidos pelas demais subtarefas. Este é um caso típico de abstração Produtor-Consumidor, normalmente utilizada em uma ampla gama de aplicações de programação concorrente.

Assim, a subtarefa de otimização é consumidora de restrições e gera novos pontos de operação e, conseqüentemente, novos estados. Tão logo novas violações de restrições sejam encontradas por outras subtarefas, a subtarefa de otimização altera seu ponto base de operação. A subtarefa geração de restrições consome estados e casos de contingências que são analisados, gerando como produto restrições de segurança para aqueles casos que apresentarem violações.

[3] Segundo este modelo, processos consomem restrições ou estados, assim como também produzem restrições ou estados.

Especificamente quanto à decomposição, há detalhes importantes em relação às subtarefas de cada um dos três programas implementados. Esses detalhes são discutidos a seguir.

Programa FPORS-I [12]

Neste programa, a subtarefa geração de restrições é realizada em conjunto com a subtarefa análise de contingências e não requer solução de PL nos processos remotos.

Um dado processo remoto recebe o estado enviado pelo *host-login* e efetua a análise da contingência, para, em seguida, gerar uma restrição de segurança do tipo sensibilidade entre fluxo e geração.

A restrição gerada por um dado processo remoto é recebida pelo *host-login* e incluída no algoritmo de otimização (PL do caso-base). Finalizado o PL do caso-base com a inclusão da nova restrição em sua base, o processo do *host-login* remete um novo estado aos demais processos. O assincronismo entre as tarefas Consumidora e Produtora é conseguido fazendo-se com que o processo do *host-login* não tenha que aguardar restrições dos processos remotos; o processo do *host-login*, na eventualidade de não receber restrição, irá gerar restrições também.

Programa FPORS-II

Neste programa, a subtarefa geração de restrições é realizada em conjunto com a subtarefa análise de contingências. A cada nova restrição, o subproblema de operação é resolvido considerando-se o remanejamento (PL de operação). Cada processo remoto, sempre que for necessário, envia ao *host-login* a restrição de segurança.

O PL do caso-base não precisará ser executado se o remanejamento for suficiente. Em caso contrário, ou seja, se o remanejamento não for suficiente, efetua-se o PL do caso-base e o novo estado obtido é remetido aos processos remotos para que estes executem análise de contingência a partir do estado mais recente.

Programa FPORS-III [26]

Neste programa, a geração de restrição requer a solução do PL relativa ao subproblema de operação. A restrição é obtida a partir dos multiplicadores simplex do PL de operação (i.e., restrição do tipo Benders). Estes cálculos são efetuados pelos processos remotos.

Um dado processo remoto analisa a contingência e, se for detectada violação, atualiza o conjunto crítico e parte para a solução do PL de operação (que leva em conta o remanejamento). Calculado este PL, caso haja "infactibilidade", é gerada uma restrição do tipo Benders. Em seguida, tal restrição é enviada ao processo do *host-login*.

Ao processo do *host-login* cabe receber a restrição e incluí-la no problema de investimento (PL do caso-base). Resolvido este PL, o processo do *host-login* trata de enviar aos processos remotos os vetores de estado e de geração atuais. De modo análogo aos demais programas, na

eventualidade de não receber restrição, o processo do *host-login* irá gerar restrições também.

Quanto à comunicação no modelo Produtor-Consumidor, os tamanhos dos vetores de estado e de restrição são respectivamente iguais ao número de barras e ao número de geradores do sistema-teste. São também transmitidos valores inteiros e reais que complementam a restrição.

O pseudo-código da Figura 4.10 ilustra como o modelo Produtor-Consumidor é implementado através da construção de duas rotinas CONSUMER() e MAKER().

```
c   Depois de Resolvido o FPO Caso-base;
c   Depois de Obtido o Ranking das Contingências;
      ...
      IF( myinst = 0 ) THEN

          call CONSUMER()        ! produz estados e consome restrições

        ELSE IF( myinst ≠ 0 ) THEN

          call MAKER()           ! consome estados e produz restrições

      END IF

c   Finaliza Processos Se Não Restar Mais Casos Críticos
```

Figura 4.10: Pseudo-código da parte intensiva do programa distribuído (modelo Produtor-Consumidor para as iterações do FPORS).

A rotina MAKER() é executada pelos processos remotos, enquanto a rotina CONSUMER() é executada no *host-login*. As Figuras 4.11 e 4.12 ilustram através de pseudo-códigos, em linguagem simbólica, os principais passos das rotinas paralelas do processo iterativo do FPORS.

Critério de Parada: Convergência Paralela Global

O critério de parada baseado em marcas, conforme explanado na seção 2.4.6 do Capítulo 2, foi estendido aos programas distribuídos de FPORS.

Nos programas distribuídos, cada processador possui uma lista privativa de casos de contingência para analisar. Dos passos das rotinas CONSUMER() e MAKER() (Figuras 4.11 e 4.12), verifica-se que, dentre os processadores alocados para executar um FPORS, o único que produz novo estado básico (i.e., que resolve problemas de investimento) é o *host-login*, ou seja, onde é chamada a rotina CONSUMER(). Portanto, o processo deste *host* incrementa de um o valor do índice i_θ sempre que novo par $(\mathbf{\Delta P^0}, \mathbf{\Delta\theta})$ é obtido; o valor corrente de i_θ é enviado aos processos remotos juntamente com o correspondente estado básico. Desta forma, o controle da obtenção de novos estados é exclusivo do processo do *host-login*.

```
      Rotina CONSUMER()
      ...
c   Envia a Lista de Contingências Críticas
  100 CONTINUE
      call ENVIA_ESTADO()                    ! envia também i_θ
c   Se Houver Restrição no Buffer Faça:
      call RECEBE_RESTRIÇÃO()
      call PL_CASO_BASE_COM_RESTRIÇÃO()      ! incrementa i_θ
c   Vá para 100
c   Se Não Houver Restrição no Buffer Faça:
c   Todos os Casos Críticos Foram Processados?
c       Em Caso Afirmativo, Testa Critério de Parada;
c       Envia Aviso aos Processos;
c       Se Não for para Parar Vá para 100;
c       Se Nem Todos os Casos Críticos Foram Processados, Prossiga
c   Pega um Caso Crítico ν
      call CONTING()                         ! calcula contingência do caso ν
c   Se Houver Violação Faça:
      call CONSTRÓI_RESTRIÇÃO()              ! resolve PL no programa FPORS-III
      call PL_CASO_BASE_COM_RESTRIÇÃO()      ! incrementa i_θ
      marca(ν) ← i_θ                         ! marca o caso
c   Se Não Houver Violação Faça:
      marca(ν) ← i_θ                         ! marca o caso
c   Vá para 100
```

Figura 4.11: Pseudo-código simplificado da rotina CONSUMER().

```
        Rotina MAKER()
        ...
c   Recebe a Lista de Contingências Críticas

  200   CONTINUE

c   Pega um Caso Crítico ν

        call RECEBE_ESTADO()          ! recebe também i_θ

        call CONTING()                ! calcula contingência do caso ν

c   Se Houver Violação Faça:

        call CONSTRÓI_RESTRIÇÃO()     ! resolve PL no programa FPORS-III

        call ENVIA_RESTRIÇÃO()

c   Se Não Houver Violação Faça:

        marca(ν) ← i_θ                ! marca o caso

c   Verifica se a Convergência foi Alcançada:
c   NÃO: Vá para 200
c   SIM: Fim
```

Figura 4.12: Pseudo-código simplificado da rotina MAKER().

Por outro lado, sempre que um processo remoto, ou o próprio processo do *host-login* (rotina CONSUMER()), calcula um caso de contingência de sua lista privativa (ou não constata violação), ele marca este caso com o valor vigente do contador i_θ. Os processos remotos, assim que esgotam suas listas privativas, informam prontamente ao processo do *host-login* o valor de i_θ mais recente com o qual suas contingências foram rotuladas. Este processo, ao receber o valor de i_θ, faz uma comparação entre este valor e o índice vigente: caso sejam coincidentes significa que a convergência paralela global foi atingida; se não coincidirem, reinicia-se o processo iterativo a partir do ponto de operação atual.

Com este critério, ao término do processo iterativo do FPORS paralelo, todas as contingências deverão estar com a mesma marca, ou seja, com valores idênticos de i_θ nas posições ν do vetor de marcas. Isto indica que todos os casos de contingência foram processados em relação ao estado básico mais recente. Este critério assegura que a otimalidade da solução não seja afetada e, também, que os processadores envolvidos na solução do problema não executem mais trabalho que o estritamente necessário para atingir a solução ótima final [43].

Capítulo 5

ANÁLISE DE SEGURANÇA ESTÁTICA DISTRIBUÍDA USANDO O MODELO Pipeline

5.1 Introdução

Este capítulo tem o objetivo de propor um novo modelo de programação distribuída, denominado *Pipeline*, adequado aos problemas de análise de segurança de redes elétricas. No cotidiano de um *EMS*, a análise de segurança realiza os cálculos mais "pesados" de forma ininterrupta. A intervalos regulares de tempo, dados oriundos do subsistema de telemedição chegam ao estimador e, após serem tratados, penetram os módulos de processamento da análise de segurança. Neste trabalho, o problema específico tratado é a análise de contingências e o cálculo de estratégia corretiva.

Na Figura 5.1, as funções de segurança indicadas pelos algarismos I e II são, respectivamente, a fase de seleção de contingências, onde são identificadas as sobrecargas em ramos, e a determinação de estratégias corretivas de sobrecargas. A estratégia corretiva será colocada em prática caso as contingências postuladas venham a ocorrer.

Em um conjunto de computadores (microcomputadores ou estações) configurado como um *Pipeline*, o processador j envia resultados ao processador $j + 1$ e recebe resultados do processador $j - 1$. Os dados ou resultados fluem através dos processadores tal como um fluido é conduzido por uma tubulação. Com as facilidades oferecidas pelo sistema de programação distribuída (o *PVM3*), cada seção do *Pipeline* é formada por um ou mais processadores, que trabalham concorrentemente para concluir suas tarefas. Cada seção do programa distribuído é especializada em determinado tipo de tarefa. As tarefas de computação intensiva, mapeadas

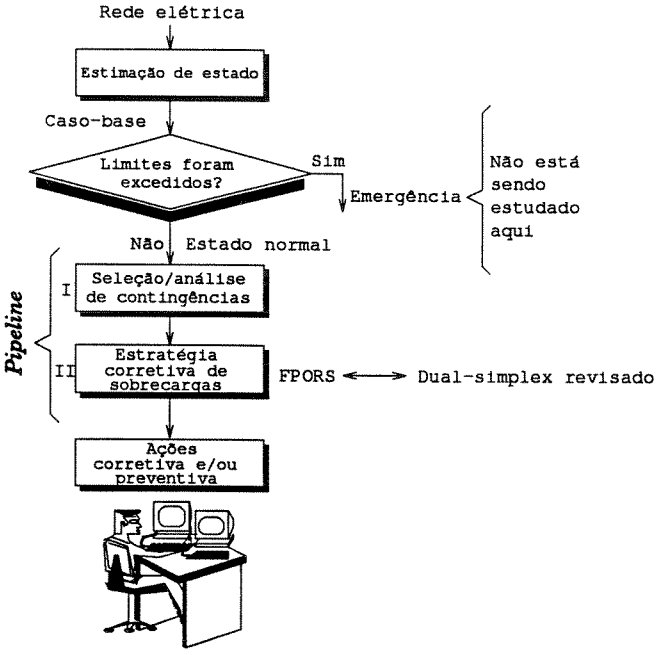


Figura 5.1: Módulos de segurança estática decompostos segundo o modelo *Pipeline*.

segundo o modelo*Pipeline*, formam um programa paralelo/seqüencial assíncrono. Cada processador, ou grupo de processadores, faz um pedaço do trabalho total operando sobre os dados que fluem de um estágio para o outro adjacente, sucessivamente.

O modelo *Pipeline*, cuja eficácia será comprovada relativamente às tradicionais metodologias de paralelização, presta-se a todo tipo de problema de múltiplos estágios de cálculo (i.e., múltiplas funções), em que diferentes bases de dados precisam ser processadas de maneira contínua e chegam periodicamente à entrada do programa. Este modelo, sem dúvida, se aplica a uma vasta gama de problemas do mundo físico e sistemas de tempo-real em geral [53]. A proposta deste modelo é a principal contribuição deste capítulo e, embora se tenha limitado a testar sua eficácia num problema típico da indústria de energia elétrica, vislumbram-se outras aplicações práticas.

Neste novo método, o problema de análise de segurança é decomposto em uma seqüência de funções: leitura e preparação de dados, seleção de contingências, avaliação de contingências para determinar estratégias corretivas, e saída de resultados. O objetivo da função seleção de contingências é identificar as contingências mais severas de uma lista de prováveis contingências, a partir de um caso-base. O caso-base é obtido da solução de um fluxo de potência e da otimização das gerações ativas com a rede intacta, respeitando-se os limites operacionais. Duas funções objetivos estão disponíveis: (1) desvio quadrático de um ponto inicial; e (2) custo de operação. Esta etapa simula as condições instantâneas da rede colhidas *in loco* e pré-processadas

pelo estimador. A seleção identifica os casos de sobrecargas e classifica as contingências em ordem de severidade. Após a obtenção do *ranking* das contingências segue a determinação de estratégias corretivas necessárias para eliminar sobrecargas causadas por uma dada contingência. Esta etapa é baseada no FPORS, supondo-se que as gerações ativas do sistema elétrico possuem capacidade de remanejamento suficiente. O valor da função objetivo do FPORS para cada contingência em separado (i.e., mínimo desvio quadrático em relação ao caso-base) é uma medida da dificuldade de correção da sobrecarga oferecida pela contingência. Este cálculo também fornece a posição dos controles ativos com a qual as violações são corrigidas. O programa transportado para o modelo *Pipeline* é o próprio programa **ranking** descrito no Capítulo 3, na seção 3.4.

As várias funções de análise de segurança são mapeadas em uma rede de estações que trabalham como um *Pipeline* com um fluxo contínuo de casos-base, conforme ilustra a Figura 5.2.

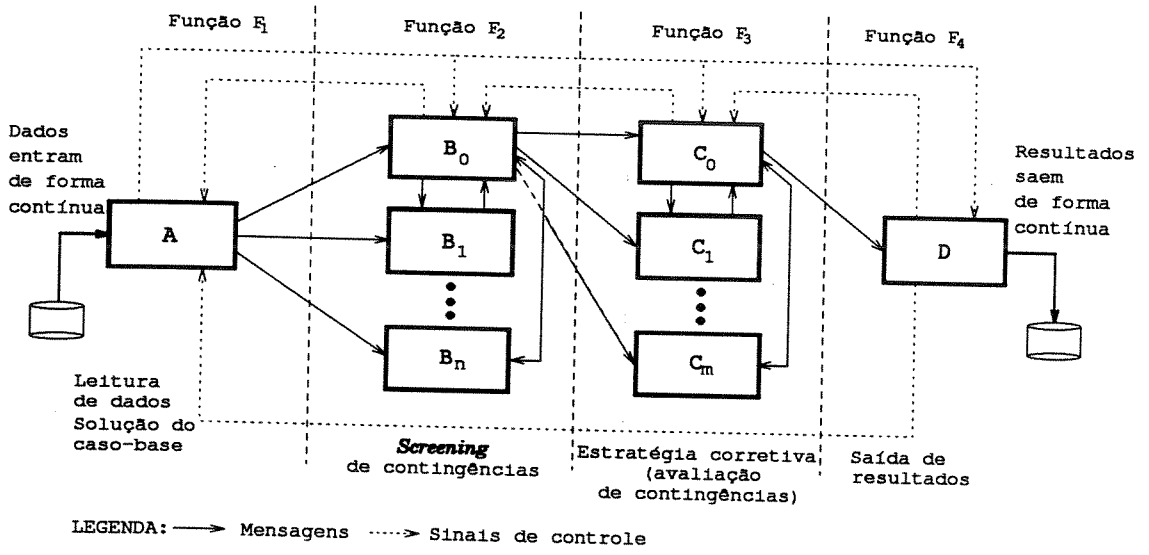


Figura 5.2: Decomposição em funções do problema de análise de segurança estática.

As funções de seleção e avaliação de contingências são decompostas usando um algoritmo assíncrono. O número de máquinas na rede associadas à seleção de contingências pode variar dinamicamente de acordo com as mudanças de condições de carga (por exemplo, causadas por variações significativas na topologia ou no tamanho da lista). O *Pipeline* também apresenta certo grau de tolerância a falha: se um processador se torna lento devido a outras tarefas executando concorrentemente, ou até mesmo indisponível, um novo processador pode ser adicionado ao *Pipeline* em tempo de execução, sem prejuízo para o programa.

Um sistema de *software* de alto nível, e de domínio público, foi usado para paralelizar o problema de análise de segurança em uma rede de estações de trabalho (e.g., rede *LAN* com estações *SunSPARC*, pertencentes ao ambiente descrito no Capítulo 4): o sistema *PVM, Parallel Virtual Machine* [52]. O *PVM*, conforme está descrito no Capítulo 4, é baseado em transferência de mensagens e permite aos programadores explorar recursos de computação distribuída através

de diferentes tipos de máquinas.

Por comparações entre implementações do mesmo programa, uma feita no modelo *Hosted* e a outra no modelo *Pipeline*, excelentes resultados foram obtidos e estão mostrados no Capítulo 6, de testes e resultados. Uma das principais dificuldades normalmente encontradas em aplicações distribuídas sobre uma rede de estações relaciona-se à perda de eficiência à medida que o número de estações aumenta. A técnica de paralelização proposta atrasa a saturação das curvas do *speedup* e assim apresenta superior eficiência e reduzidos tempos de computação.

O capítulo é organizado como segue. Primeiramente, na seção 5.2, quatro modelos de programação concorrente são discutidos. Em segundo lugar, na seção 5.3, os detalhes das implementações nos modelos *Hosted* e *Pipeline* são apresentados. São discutidas ainda nesta seção as medidas de desempenho adequadas a esta modalidade de programação. Na última seção (a 5.4) são descritas as propriedades dinâmicas do programa *Pipeline*.

5.2 Modelos de Programação Concorrente

Esta seção descreve quatro técnicas alternativas de decomposição para análise de segurança estática: duas delas são métodos de decomposição por dados, denominadas modelos *SPMD* e Assíncrono, e as outras são métodos de decomposição por funções, designadas por modelos *Hosted* e *Pipeline*. O modelo *Pipeline* é a nova técnica sugerida neste capítulo; o modelo *Hosted* é uma metodologia de paralelização tradicional que é analisada neste capítulo (que opera por batelada) e foi implementada para fins de comparação com o modelo proposto. Os outros dois modelos são apresentados apenas para o propósito de discussões.

5.2.1 Técnica de Decomposição por Dados

Nesta técnica, a carga de trabalho computacional de uma lista de contingências (seleção e avaliação) é dividida entre os processadores da máquina paralela criada com o *PVM3*. Há basicamente duas diferentes maneiras de fazer a divisão, o *SPMD* e o Assíncrono. Estes dois modelos podem ser usados como blocos construtores de modelos mais complexos, ou como modelos independentes. O modelo Assíncrono em particular é uma parte importante do modelo *Pipeline* proposto na seção 5.3.2.

Modelo *Single Program Multiple Data (SPMD)*

Múltiplas estações executam o mesmo pedaço do código fonte, com cada estação trabalhando sobre um subconjunto, previamente definido, do conjunto de dados. Um exemplo que dá uma boa "imagem" deste modelo é a somatória em paralelo de uma seqüência de valores numéricos.

Por exemplo, na área de sistemas de potência, dada uma lista de N_c severas contingências para serem analisadas usando um programa de fluxo de potência AC, e dado p processadores, cada processador analisaria N_c/p contingências (vide seção 4.5.1, capítulo anterior). Assumindo-se que cada contingência requer o mesmo esforço computacional para ser calculada, este modelo apresenta um comportamento frouxamente sincronizado.

Modelo Assíncrono

Cada processador retira casos de um *pool* de contingências gerenciado por um dos processadores. Por exemplo, dado um conjunto de N_c severas contingências, cada processador, executando um código de análise de contingências, retira uma a uma contingência da lista (i.e., do *pool* de contingências). Neste modelo, assim que um processador termina o seu cálculo, ele solicita prontamente outro caso para calcular. O processo pára quando o *pool* estiver vazio; neste momento, o número de contingências que terá sido analisado por cada processador não necessariamente será o mesmo, uma vez que a avaliação de diferentes contingências pode exigir tempos de computação bastante diferentes. A Figura 5.3 ilustra este modelo através de um perfil de execução de três processos concorrentes.

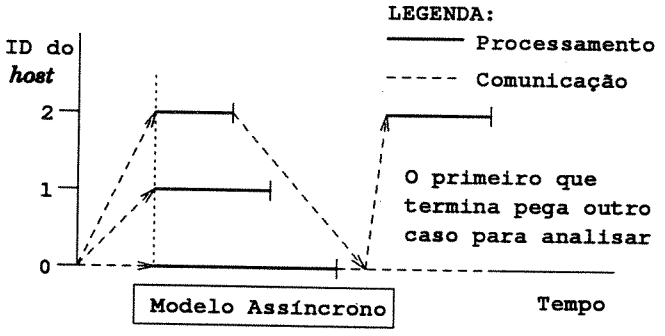


Figura 5.3: Perfil de execução para o modelo Assíncrono (o eixo horizontal representa o tempo de computação).

5.2.2 Técnica de Decomposição por Funções

Nesta técnica, a carga computacional de análise de segurança estática, isto é, as funções de entrada/saída de dados e resultados, seleção de contingências e determinação da estratégia corretiva das sobrecargas, é dividida entre diferentes grupos de processadores da máquina paralela virtual. Duas alternativas de implementações são estudadas, implementadas e testadas: os modelos *Hosted* e *Pipeline*.

Modelo *Hosted*

Uma série de pedaços paralelo/seqüencial de códigos são evocados por um código *master*

que executa em um processador *host* (i.e., *host-login*). Os códigos paralelos podem ser dos tipos Assíncrono ou *SPMD*. Por exemplo, no caso de análise de segurança, o *host* pode evocar: (1) um pedaço seqüencial do código para leitura dos dados e solução do caso-base; (2) um pedaço paralelo do código para seleção de contingências; (3) um pedaço seqüencial do código para ordenação das contingências em termos de severidade das sobrecargas; (4) um outro pedaço paralelo do código para avaliação das contingências e determinação de estratégia corretiva de sobrecargas (gerando então mais um índice, que mede a dificuldade de correção da violação); (5) um pedaço seqüencial do código para uma nova classificação das contingências segundo o novo índice PI_{MDQ}; e (6) um pedaço seqüencial do código para escrever em arquivo os resultados. Códigos de seleção de contingências e avaliação de contingências podem usar o modelo frouxamente acoplado *SPMD* ou o modelo Assíncrono, ambos descritos acima.

Modelo Heterogêneo *Pipeline* (*MPMD*)

Múltiplos códigos paralelos, executando em múltiplas estações de trabalho, transferindo dados (ou resultados) entre si e através das seções como se fosse um sistema físico de transporte de matéria (por exemplo, gasoduto, oleoduto, etc.), conforme está ilustrado na Figura 5.2. Este modelo é similar ao modelo *Hosted*, exceto que agora as tarefas desempenhadas pelo computador *host* são mapeadas como um *Pipeline* cujas seções se compõem de grupos de processadores. No caso de análise de segurança, no primeiro estágio do *Pipeline* tem-se um único processo executando em um único processador e perfazendo as operações de leitura de dados e solução do caso-base. No segundo estágio, tem-se um pedaço de código paralelo responsável pela seleção de contingências: há um processo pai e um certo número de processos filhos; o processo pai, além de calcular casos de contingências, sincroniza todos os processos filhos para receber dados do estágio anterior; ativa os processos filhos, ordena as contingências mais severas, e é responsável pelas comunicações com outros estágios. O *screening* de contingências envolve a solução rápida e aproximada (e.g., fluxo de potência DC incremental) de uma lista de prováveis contingências e pode ser codificado seja como um *SPMD* ou como um modelo Assíncrono. O terceiro estágio faz a determinação da estratégia corretiva com base na solução de FPORS (algoritmo dual-simplex revisado de Stott), com a rede retratando cada caso de contingência da lista proveniente do segundo estágio. O resultado do terceiro estágio é apresentado de forma ordenada (decrescente de severidade) e indica a decisão a tomar no evento de cada uma das contingências postuladas. O código paralelo deste estágio possui o mesmo aspecto daquele usado no segundo estágio, embora em função e natureza sejam completamente distintos (solução de PL para cada condição de distúrbio). No quarto e último estágio do *Pipeline* tem-se um único processo, possivelmente executando no mesmo processador que executa o primeiro estágio.

5.3 Detalhes das Implementações

Esta seção descreve as implementações distribuídas dos modelos *Hosted* e *Pipeline* em uma rede de estações provida com o *PVM3*, em sua versão mais recente. Uma máquina paralela virtual com p nós é inicialmente definida pela aplicação *PVM3*: nó−0 (ou *host-login*) que trabalha

como um gerente (ou processo *master*) e nós 1 até $p - 1$ que trabalham como servidores.

5.3.1 Programa ranking no Modelo *Hosted*

Dois pedaços de código fonte executam concorrentemente na máquina paralela virtual e se comunicam através da rede *FDDI* ou *Ethernet*: um código *master* executa em uma estação (nó–0) e aloca (*spawning* = desova) cópias dos códigos *slave* em todas as estações incluindo o próprio nó–0 (i.e., nós 0, 1, $\cdots$, $p - 1$); a UCP do nó–0 é compartilhada pelo processo *master* e por uma tarefa *slave*, enquanto todas as outras estações executam as demais tarefas *slave*. O *master* é responsável pela execução de leitura de dados e opções, e pela inicialização de todas as outras tarefas (enviando números de identificação de processadores e tarefas). Um grupo dinâmico (de acordo com a nomenclatura do *PVM3*) compreendendo p tarefas idênticas é criado pelo *master*.

O *master* inicialmente toma um "caso", isto é, um caso-base com dados típicos do fluxo de potência, com uma lista de contingências e com dados de geradores para o PL, e aloca p cópias das tarefas *slave* (uma para cada estação incluindo o nó–0). Adicionalmente, o nó–0 é responsável pelo cálculo do caso-base (fluxo de potência inicial e FPO da rede intacta) e pelo gerenciamento da lista de contingências simples e múltiplas, a partir da qual as tarefas *slave* retiram casos para analisar.

Cada tarefa *slave* recebe dados e começa executando suas rotinas privativas de fluxo de potência e FPO da rede intacta (vide Figura 4.4 do capítulo anterior), o qual produz a solução do caso-base[1]; após obter esta solução, a tarefa *slave* retira um caso de contingência do *master* e processa o fluxo de potência no modo *screening* (baseado em um modelo linearizado de fluxo de potência com técnicas de esparsidade de vetores e matrizes); tarefas *slave* retiram novos casos de contingências do processo gerenciador à medida que concluem seus cálculos, de maneira assíncrona (vide Figuras 4.8 e 4.9); este procedimento segue até nenhuma contingência restar na lista. O *master* recebe os índices de severidade (neste caso, índices PI_{MW}) enviados pelas tarefas *slave*, ordena as contingências e seleciona as mais severas para posterior análise (de uma lista reduzida).

Basicamente, o mesmo procedimento assíncrono usado para a seleção ou *screening* é então repetido para a fase de avaliação de contingências. As duas principais diferenças são: agora somente uma lista reduzida, contendo as mais severas contingências, é avaliada, e um FPO, com a matriz de rede (i.e., os fatores de $[B']$) devidamente atualizada, é resolvido para cada contingência. A função objetivo deste FPO é o desvio quadrático em relação ao ponto do caso-base, que dá uma medida da intensidade da correção das gerações exigida para sanar o efeito da contingência e tornar o cenário pós-contingência viável. O valor do desvio quadrático, resultante da solução do FPO, é tomado como índice que é então usado para reordenar as contingências

[1]Todos os processos resolvem repetidamente o fluxo inicial e o FPO da rede intacta, a partir dos dados do caso-base. Isto é quase tão rápido quanto a tarefa *slave* do nó–0 sozinha resolver o caso-base e então comunicar os resultados às tarefas *slave* remotas.

desta lista reduzida. Nesta fase, mais que durante a fase de *screening*, o método assíncrono de alocação de carga computacional pode ser crítico, desde que o esforço requerido no cálculo do FPO de diferentes contingências pode variar amplamente. Na última fase, o *master* faz a saída dos resultados obtidos das tarefas *slave* e das suas próprias operações.

Notas:

a) A Figura 5.4 mostra um perfil de execução paralela do programa ranking para o modelo *Hosted*, indicando que a seleção e a avaliação de contingências são processadas em paralelo, enquanto entrada/saída de dados e de resultados, e as ordenações das listas de contingências são efetuadas seqüencialmente, o que eventualmente limita o *speedup* que pode ser alcançado usando este método.

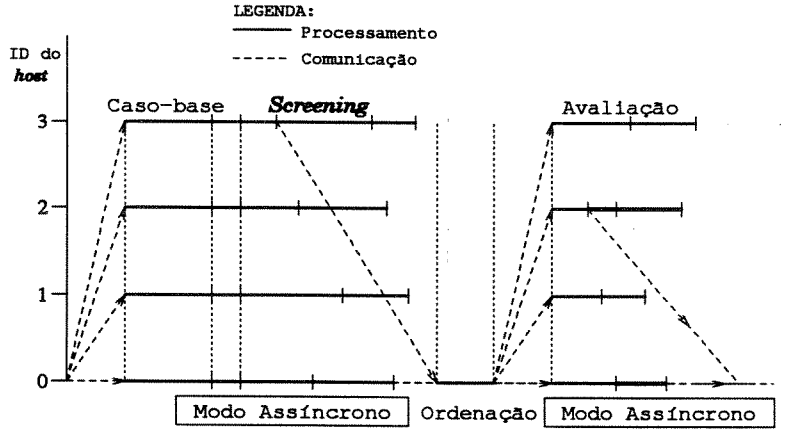


Figura 5.4: Perfil de execução do programa distribuído segundo o modelo *Hosted*.

b) A Figura 5.5(a) ilustra como o modelo *Hosted* processa uma seqüência de casos-base, enquanto a Figura 5.5(b) ilustra as operações do modelo *Pipeline* com configuração mínima, isto é, 3 máquinas: *I/O* e caso-base, seleção, e avaliação.

5.3.2 Programa ranking no Modelo *Pipeline*

Este modelo desempenha as mesmas funções do modelo *Hosted*, exceto que agora estas funções estão concatenadas em um *Pipeline* conforme está ilustrado na Figura 5.2. O principal benefício é o alto grau de utilização de UCP que culmina em *throughputs*[2] superiores. Assim como no modelo *Hosted*, o problema de análise de segurança é decomposto em leitura de dados,

[2] *Throughput* é uma medida de desempenho usual em programas distribuídos que são executados ininterruptamente submetidos a um fluxo de dados [49]. Na presente aplicação, o *throughput* observado em um estágio do programa seria expresso em número de casos de contingências processados na unidade de tempo.

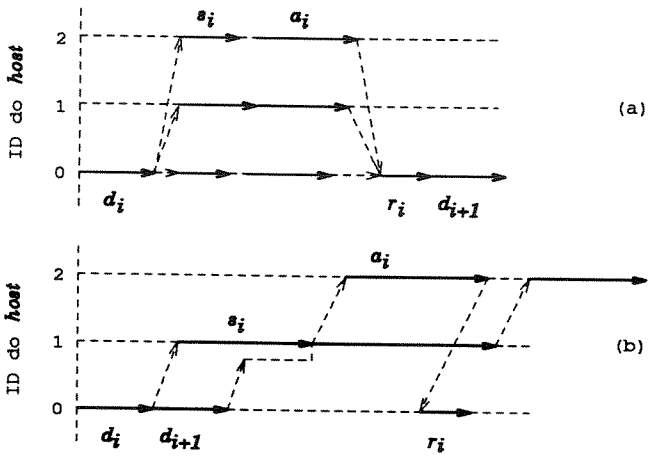


Figura 5.5: Comparação entre os modelos (a) *Hosted* e (b) *Pipeline* para uma seqüência de casos-base; d_i representa os dados de entrada e solução do caso-base; s_i representa o *screening*; a_i representa avaliação; r_i representa a saída de resultados (para simplificar, a ordenação não é mostrada explicitamente, e em (b) *screening* e avaliação de contingências são efetuados por processadores individuais, 1 e 2 respectivamente).

solução do caso-base (que compreende o cálculo do fluxo de potência inicial e o FPO da rede intacta), seleção de contingências, avaliação das contingências para determinação de estratégias corretivas, e saída de resultados em arquivo. Também como antes, as funções de seleção e avaliação de contingências são subdivididas em tarefas paralelas usando um algoritmo assíncrono. O modelo é implementado como um grupo dinâmico de acordo com a terminologia do *PVM3*, compreendendo quatro tipos distintos de processos: tipo A para entrada de dados e cálculo do caso-base, tipo B para *screening*, tipo C para avaliação de contingências, e tipo D para saída de resultados (vide Figura 5.2). Os processos dos tipos B e C normalmente possuem múltiplas cópias, cada cópia executando em uma estação:

1. O processo seqüencial A administra a entrada de dados, envia dados a outros processos, e calcula a solução do caso-base (d_i na Figura 5.5(b)).
2. Os processos assíncronos paralelos B recebem dados do processo A, faz seleção de contingências (s_i na Figura 5.5(b)), identifica condições de ilhamento, calcula índice de severidade para cada contingência, e ordena as contingências de acordo com os índices obtidos.
3. Os processos assíncronos paralelos C recebem uma lista das mais severas contingências produzidas pelos processos B, calculam FPO com a matriz de rede $[B']$ devidamente alterada, minimizando o desvio quadrático do ponto básico de operação, detectam contingências que levam a condições de "infactibilidade", reordena as contingências, e envia resultados ao processo D (a_i na Figura 5.5(b)).

4. O processo seqüencial D gerencia a saída de resultados à medida que são produzidos pelo *Pipeline* (r_i na Figura 5.5(b)). Processos A e D podem ou não compartilhar uma mesma UCP, conforme ilustra a Figura 5.6.

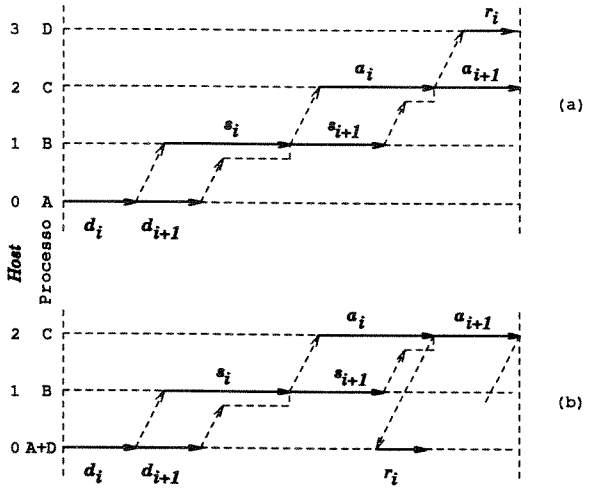


Figura 5.6: Duas variantes do modelo *Pipeline*: (a) diferentes processadores para entrada e saída de dados e resultados; (b) entrada/saída no mesmo processador.

Notas:

a) As implementações dos processos dos tipos B e C como parte de um grupo dinâmico possibilitam variar o número de cópias *slave* para acomodar mudanças nas condições de carga computacional (é possível através de ferramentas do *UNIX* monitorar variáveis, como a taxa de utilização da UCP, e a própria aplicação *PVM3* decidir, em tempo de execução, sobre mudanças na configuração ou nos processos); também, tolerância a falha é suportada até um certo grau (a substituição de um processo *slave* foi implementada no modelo *Pipeline* e teve bom desempenho e comportamento confiável).

b) Para alcançar máxima eficiência, os vários estágios do *Pipeline* devem ser "casados" em *throughput*. O estágio mais lento do *Pipeline* define seu máximo *throughput*. Para manter o balanceamento da carga dentro de um estágio, a técnica assíncrona descrita anteriormente foi adotada (tarefas *slave* retiram casos de contingências de um *pool* de contingências), conforme foi explanado na seção 4.5.1. Para manter o balanço (i.e., o equilíbrio) entre os estágios, comunicação e cargas computacionais devem ser uniformemente distribuídas entre os estágios, e isto é feito associando, e re-associando dinamicamente se necessário for, o apropriado número de processadores para os estágios B e C do *Pipeline*.

c) A coordenação do *Pipeline* é conseguida através de trocas de mensagens (i.e., sinais) entre os processos pai dos estágios B e C, e entre os processos A e D. Na Figura 5.2 chama-se

a atenção do leitor para as linhas tracejadas com setas, que representam os sinais de coordenação. Por exemplo, o processo A inicia lendo um novo caso-base e, sucessivamente, transmite os dados lidos aos processos B (com PVMFBCAST()); em seguida, efetua cálculos do caso-base e, mediante a recepção de um sinal oriundo do processo pai do estágio subseqüente (B), envia os resultados dos seus cálculos recentes aos processos B. Tão logo A termine a transmissão dos seus resultados básicos, ele estará pronto para ler uma nova base de dados. O mesmo tipo de comunicação é utilizada na coordenação das atividades dos estágios C e D, com a diferença de que a interação será entre B e C e, depois, entre C e D. Ou seja, somente estágios vizinhos interagem durante o processamento intensivo. Esta propriedade do programa garante independência aos outros estágios, permitindo que o processos dos estágios operem simultaneamente sobre bases de dados distintas.

d) A principal vantagem do modelo *Pipeline* proposto sobre o *Hosted* é o *throughput* mais alto, que se traduz em melhores níveis de utilização dos processadores alocados para a execução do programa e, conseqüentemente, menores tempos de computação. O preço pago por estes benefícios é o requisito de dispor de pelo menos três máquinas para iniciar a execução do *Pipeline*.

e) Implementações distribuídas na forma de *Pipeline* incrementam o *throughput* e não atrasam a entrada de dados e a saída de resultados. Duas são as razões apontadas para explicar esta *performance*: o assincronismo entre estágios (que são seqüenciais) e o assincronismo entre tarefas paralelas; e a superior capacidade de aproveitamento do recurso computacional. A Figura 5.7 ilustra a comparação dos modelos *Hosted* e *Pipeline* quanto às diferenças no grau de utilização de UCP, por meio de dois diagramas espaço-tempo. A partir desta figura fica claro que, submetidos a um fluxo de dados, o *Pipeline* apresenta melhor índice de aproveitamento de "energia computacional" (UCP×tempo) que o outro modelo; cada bloco representa um estágio com processos concorrentes.

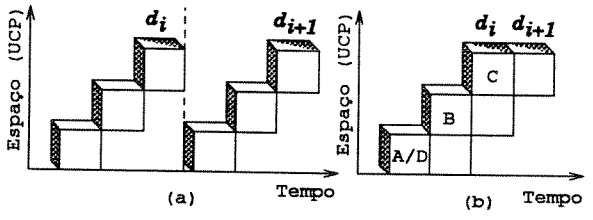


Figura 5.7: Diagrama UCP×tempo: (a) modelo *Hosted*; (b) modelo *Pipeline*; d_i representa o caso-base i e d_{i+1} o caso-base seguinte.

Da Figura 5.7(a), observa-se que, no modelo *Hosted* uma nova base de dados (d_{i+1}) somente tem seu processamento iniciado quando o último estágio finaliza a saída de resultados (relativa a d_i), e durante o tempo em que um estágio está processando os demais ficam ociosos. Conclui-se desta observação que, submetido a um fluxo de dados, o *Hosted* tem seus estágios trabalhando seqüencialmente. No modelo *Pipeline*, assim que um estágio termina suas tarefas, e repassa os resultados ao estágio seguinte, ele estará livre para trabalhar sobre uma nova base de dados.

Medidas de Desempenho

Em processamento paralelo é comum avaliar desempenhos de aplicações através de *speedup* e eficiência, conforme definidos na seção 4.3. Por outro lado, em aplicações de tempo-real cada conjunto de dados deve ser processado dentro de um máximo tempo com a finalidade de cumprir prazos. Neste sentido, *speedup* não é uma figura de mérito fundamental de sistemas paralelos (i.e., a combinação de um algoritmo paralelo com a arquitetura no qual é implementado). Isto é especialmente verdadeiro quando se utiliza redes heterogêneas de computadores, a aplicação está sujeita a um fluxo regular de dados e há preocupações com relação à tolerância a falha.

Portanto, fugindo ao comportamento padrão para avaliar desempenhos, os dois paradigmas concorrentes são comparados quanto ao tempo de resposta quando submetidos a dez bases de dados (quantidade escolhida arbitrariamente), conforme está mostrado no Capítulo 6, de testes e resultados. A comparação dos tempos de computação dos dois modelos, *Hosted* versus *Pipeline*, realmente mostra que, nas condições testadas, para a maior parte das configurações a supremacia do *Pipeline* é verificada. A economia de tempo trazida pelo novo modelo é bastante significativa, podendo chegar em alguns casos a praticamente 60%.

Ambiente Utilizado

Para testar as implementações do programa **ranking** nos modelos *Pipeline* e *Hosted* foram utilizadas estações interconectadas através de uma rede de fibras ópticas (*FDDI*) e estações interligadas pelo meio *Ethernet*. A Figura 5.8 ilustra uma disposição possível de estações, que lembra o esquema mostrado na Figura 5.2.

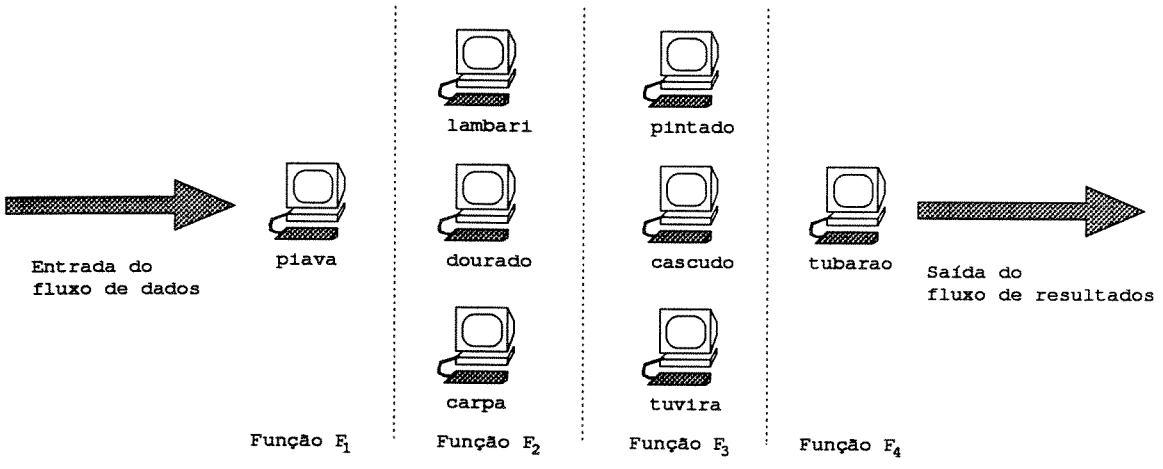


Figura 5.8: Uma possível configuração de estações para executar o programa ranking de acordo com a decomposição funcional *Pipeline*.

Os resultados obtidos para os modelos *Hosted* e *Pipeline*, para diversas configurações de máquinas, estão mostrados no Capítulo 6.

5.4 Configuração Dinâmica: Tolerância a Falha e Balanceamento de Carga

Um aspecto importante nas implementações elaboradas para executar em redes de computadores é a tolerância a falha. A saída ou a resposta lenta de uma estação da máquina paralela compromete a eficiência da operação global. Para aplicações de tempo-real, o algoritmo deve incorporar mecanismos de prevenção desses eventos. Um sistema com a capacidade de continuar funcionando, talvez de forma degradada, face à ocorrência de falhas é chamado de tolerante a falhas. Esta faculdade torna a aplicação confiável e permanentemente disponível [38,61].

Para valorizar as virtudes do *Pipeline* já apontadas no trabalho, foram incorporadas ao programa então confeccionado as propriedades dinâmicas do sistema *PVM3*. Estas propriedades permitiram a construção de um programa tolerante a falha de um *host* onde esteja executando um código *slave*. Ou seja, a qualquer instante durante a execução da computação intensiva no programa **ranking**, modelo *Pipeline*, a ocorrência de falha de uma máquina pode ser "sentida" pelo *PVM3* e, de maneira quase inteligente, a aplicação será capaz de sanar este defeito com a pronta substituição por uma máquina disponível, sem prejuízo dos cálculos em curso.

No cotidiano de um *EMS*, a carga de trabalho a ser executada pelos *hosts* dedicados à análise de segurança pode variar em função de mudanças significativas na topologia da rede elétrica, ou da eventual necessidade de processar listas mais longas de contingências. Para evitar atrasos indesejáveis na apresentação de diagnósticos de segurança, a aplicação pode variar dinamicamente o número de computadores da máquina virtual de modo a adequar a aplicação ao novo regime de operação. Esta propriedade visa manter constante a eficiência da aplicação (i.e., isoeficiência), mesmo sob condições anormais de carga computacional.

As implementações da tolerância a falha e do balanceamento da carga só são possíveis graças à existência de rotinas especializadas do *PVM3*, que permitem a programação de aplicações com a opção de configurar e reconfigurar a máquina paralela virtual de forma dinâmica em tempo de execução.

A Figura 5.9 ilustra trechos de código do processo gerente (estágio A) que contêm em linhas gerais as instruções necessárias à implementação da tolerância a falha de *host*. Com relação à Figura 5.9 algumas explicações adicionais são necessárias. As rotinas FAULT() e DYNAMIC() são implementações feitas pelo programador da aplicação: FAULT() simula a perda de uma máquina eliminando-a deliberadamente da configuração (PVMFDELHOST()); isto equivale a desconectar a estação do resto da rede; DYNAMIC() faz o controle de máquinas disponíveis e de máquinas ocupadas com processos e o tipo de processo (do estágio B ou do C); adiciona um novo *host* à configuração e nele aloca o processo afetado pela falha. Depois de ter criado o processo correto no novo membro da máquina virtual, DYNAMIC() alimenta-o com os dados atuais para habilitá-

lo a assumir casos de contingências tal como os outros do mesmo estágio. Após estas operações, as rotinas de tolerância a falha estarão prontas para enfrentar outra ocorrência semelhante.

```
c   Programa Gerente do Estágio A
        ...
c   Requisita Notificação dos Processos PVM sobre Falha de Host

        call PVMFNOTIFY( PVMHOSTDELETE,msg911,NPROC,tids,iflag )
        ...

c                          CONFIGURAÇÃO DINÂMICA (INÍCIO)

c   Simula a Falha de um Host Escolhido Aleatoriamente
        call FAULT()
        ksh = 1

c   Aguarda Notificação sobre a Ocorrência de Falha de Host

     99 call PVMFNRECV( -1,msg911,knm )

c   Se Houver Notificação Faça:
c       Leia a Notificação
c       Corrige a Falha e Aloca Processo no Host Substituto

        call DYNAMIC()
        ksh = 0

c   Se Não Houver Notificação Faça:
c       Se ksh = 1 Verifique o Buffer em 99
c       Em Caso Contrário, Prossiga

c                          CONFIGURAÇÃO DINÂMICA (FIM)
        ...
```

Figura 5.9: Trechos principais dos códigos da implementação de tolerância a falha.

O trecho do código demarcado na Figura 5.9 é repetidamente executado durante a computação intensiva. Ressalta-se que não há necessidade de qualquer instrução nos processos remotos, o sistema *PVM3* providencia os mecanismos internos de comunicação e controle. A partir dessa implementação que previne a falha de um *host*, outras rotinas de tolerância a falha podem ser elaboradas com relativa facilidade, por exemplo, com salvaguardas para *deadlock* de processos *slave*, inclusão de restrições de tempo de execução (comunicação com *time-out*), etc.

Balanceamento de carga e tolerância a falha são importantes requisitos de funções de análise de rede nos modernos Centros de Supervisão e Controle. Para o futuro almeja-se que os programas, mais que simples executores de operações aritméticas, sejam capazes de tomar decisões sobre suas próprias tarefas e, para tanto, deverão ser dotados de algoritmos inteligentes de gerenciamento e autocontrole.

Capítulo 6

TESTES E RESULTADOS

6.1 Introdução

Este capítulo descreve os sistemas elétricos, as condições de testes, os ambientes computacionais utilizados em uma série de estudos de casos, bem como os resultados obtidos com os programas digitais desenvolvidos nesta tese. A série de estudos de casos foi especialmente projetada para: (1) demonstrar a funcionalidade e a robustez das implementações; (2) comprovar expectativas de desempenho criadas ao longo do texto; (3) mostrar a qualidade dos resultados obtidos das implementações, em particular para sistemas elétricos reais; (4) demonstrar o superior desempenho do modelo *Pipeline* sobre o *Hosted*, tendo em vista os requisitos de aplicações *on-line* para Centros de Controle; (5) apresentar e discutir as virtudes e as deficiências do sistema de programação/processamento *PVM3* baseado em estações de trabalho interligadas como uma *LAN*; e (6) comparar os desempenhos das implementações distribuídas dos programas de fluxo de potência ótimo com restrições de segurança, com e sem capacidade de redespacho pós-contingência.

Breve Recapitulação

As implementações computacionais desenvolvidas nesta tese referem-se à análise de contingências e ao alívio das violações de limites de fluxos em ramos (sobrecargas em $[MW]$) através do remanejamento dos controles ativos; são elas:

(i) fluxo de potência ótimo com restrições de segurança sem capacidade de remanejamento pós-contingência; este programa é denominado de FPORS-I e é essencialmente a implementação estudada em [43];

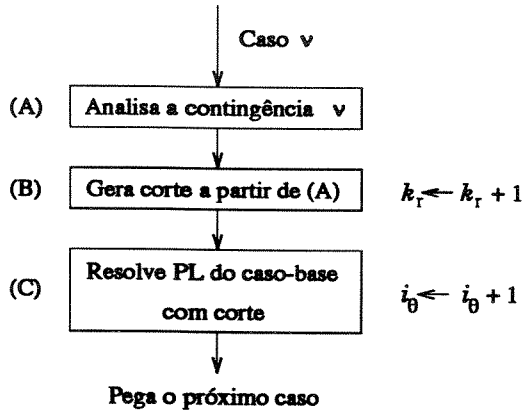


Figura 6.1: Fluxograma simplificado do processo iterativo do FPORS-I.

(ii) fluxo de potência ótimo com restrições de segurança com capacidade de remanejamento pós-contingência, que emprega a técnica sugerida por Stott *et al.* para construir o corte (i.e., a restrição de segurança) pela máxima violação originada da saída de um ou mais ramos do sistema de transmissão; esta técnica foi estendida para $\Delta \neq \mathbf{0}$; este programa é denominado de FPORS-II;

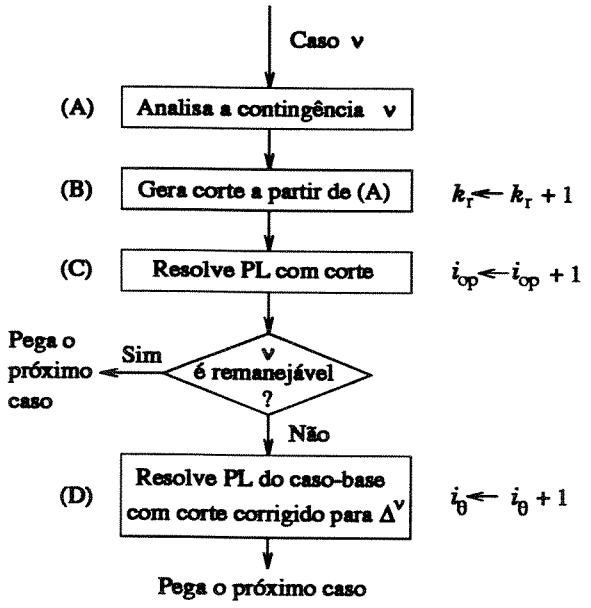


Figura 6.2: Fluxograma simplificado do processo iterativo do FPORS-II.

(iii) fluxo de potência ótimo com restrições de segurança com capacidade de remanejamen-

to pós-contingência, que utiliza decomposição de Benders para separar o problema de FPORS em dois: um nas variáveis do caso-base, e o outro nas variáveis pós-contingência. Nesta abordagem, ao contrário das duas anteriores, os cortes são gerados por multiplicadores simplex da solução dos subproblemas de contingência que resultem infactíveis. Esta implementação é denominada de FPORS-III;

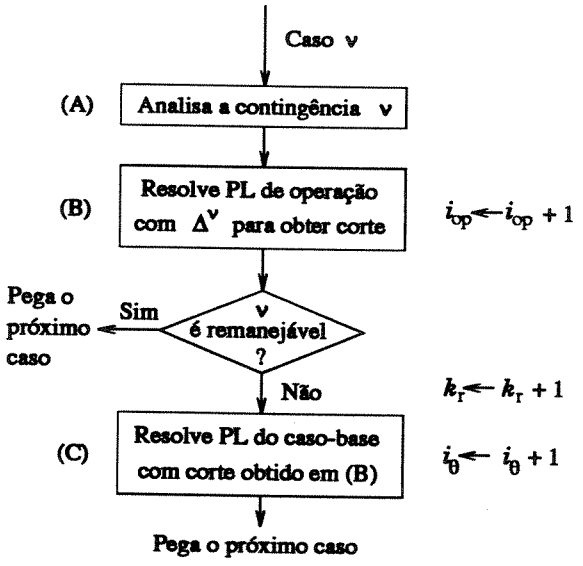


Figura 6.3: Fluxograma simplificado do processo iterativo do FPORS-III.

Nos fluxogramas ilustrados nas Figuras 6.1, 6.2 e 6.3 (que são simplificações das Figuras 2.6, 2.7 e 2.9), os contadores k_r, i_θ e i_{op} são:

- k_r é o contador de restrições de segurança geradas;
- i_θ é o contador de iterações do caso-base (PL do caso-base ou de investimento);
- i_{op} é o contador de iterações de operação (PL de operação).

Estas definições são importantes para auxiliar a compreensão das tabelas de resultados mostradas neste capítulo.

(iv) análise de contingências em duas camadas de processamento: na primeira, após o cálculo do caso-base, é feito o *screening* das contingências determinando-se os estados pós-distúrbio aproximados para verificação e construção de um *ranking* de sobrecargas em ramos (índice PI_{MW}); na segunda, a partir da lista ordenada das contingências, é efetuado o cálculo do PL para a rede alterada de acordo com a mudança topológica correspondente à contingência simulada, com função objetivo desvio quadrático; neste processamento busca-se determinar o remanejamento dos controles ativos que é necessário para corrigir a sobrecarga da contingência individualmente (índice PI_{MDQ}). Com este programa deseja-se demonstrar a coerência entre a classificação das contingências obtida com o índice PI_{MW}

e a classificação obtida a partir do índice PI_{MDQ}. Esta técnica foi proposta no Capítulo 3, seção 3.4, e o programa é batizado com o nome **ranking**.

Todas as implementações foram elaboradas inicialmente para execução em máquinas monoprocessadoras, ou seja, como programas seqüenciais; posteriormente foram paralelizadas com base no sistema *PVM3* para executar numa rede heterogênea de computadores.

Sistemas Testados e Condições Gerais dos Testes

Os testes descritos neste capítulo estão divididos em três seções. A seção 6.2 apresenta os testes e os resultados dos programas de FPORS (que são FPORS-I, FPORS-II e FPORS-III) realizados na forma seqüencial em uma estação. Nesta seção, os programas são testados com a finalidade de mostrar diferentes aspectos dos algoritmos implementados, especialmente no que se refere à velocidade, número de iterações, quantidade de restrições geradas para alcançar o ponto ótimo-seguro, e a exatidão das soluções. São estudadas também as diferenças de desempenho dos computadores para esta categoria de função de análise de rede. Na seção 6.3 são apresentados os resultados dos programas de FPORS distribuídos num conjunto de estações (i.e., uma máquina paralela virtual). Nesta seção, o objetivo é comparar desempenhos em situações variadas caracterizadas pelas dimensões dos sistemas testados, pelos tamanhos das listas de contingências e os ambientes de processamento com os meios de comunicação *Ethernet* e *FDDI*. A seção 6.4 apresenta um estudo comparativo dos modelos *Pipeline* e *Hosted* para a implementação distribuída do programa **ranking**. O objetivo principal das comparações é mostrar como os tempos de computação variam com o número de processadores no *Pipeline* e com as cargas computacionais (número de barras e número de contingências). Nesta seção, ainda são mostrados resultados de testes com este programa que visam validar a utilização dessa técnica nos modernos Centros de Supervisão e Controle, seja como uma implementação seqüencial, ou distribuída num conjunto de máquinas. As propriedades dinâmicas do *PVM3* são ilustradas conforme estão implementadas no *Pipeline*: é simulada a falha de um *host* durante a execução.

Os sistemas elétricos testados compreendem desde redes pequenas como o $IEEE$-30, redes médias como o $IEEE$-300, até redes elétricas de transmissão do Sistema Interligado do Brasil (de até 750 $[kV]$), consideradas como sistemas de grande porte (por exemplo, a rede BR-1663 de 1663 barras e 2349 ramos). As listas de contingências são de tamanhos diversificados e possuem saídas simples e saídas múltiplas simultâneas de ramos; a composição e o tamanho da lista de cada sistema-teste são escolhidos em função do efeito que se deseja investigar no caso em estudo. Alguns dados de geradores (como custos e limites), e dados de limites de fluxos que não estejam disponíveis nos bancos de dados originais *PECO* ou *CDF* foram criteriosamente estimados, de modo a suprir essas faltas.

As redes de computadores, *Ethernet* e *FDDI*, foram utilizadas nas condições livre e ocupada. Sendo que, ocupada significa compartilhar os recursos, UCP e meio de comunicação, com processos de outros usuários, em condições de um dia normal de trabalho no laboratório. A condição livre corresponde à ausência total de processos de outros usuários, com os programas execu-

tando em ***background*** geralmente durante à noite. Contudo, é importante frisar que, mesmo durante a condição livre, a rede encontra-se conectada ao resto do mundo, estando então sujeita ao recebimento de mensagens eletrônicas ou a interferências de outros sinais no decorrer de um processamento. Os tempos mostrados nas tabelas e figuras deste capítulo são valores médios obtidos em uma série de cinco medidas (ou simulações); os tempos transcorridos normalmente dados em segundos incluem *I/O*, comunicação, e tempos de processamento e correspondem aos mesmos tempos que seriam obtidos se fosse empregado um cronômetro. Todos os programas foram compilados com o f77 com a opção de otimização **-dalign -O2**.

Os ambientes de processamento utilizados nos testes estão detalhados nas Tabelas 6.1 e 6.2. Na Tabela 6.2, "outras" representa as demais estações cujo modelo está indicado.

Tabela 6.1: Características das estações de alto desempenho *SunSPARC ULTRA* utilizadas como processadores seqüenciais.

Ambiente/ Rede	Nome da estação	Modelo	Características: *RAM*, *clock*, etc.			
			[*MBytes*]	[*MHz*]	[*MIPS*]	[*MFLOPS*]
Ethernet	piranha	*SPARC ULTRA1* 170	128	167	341,0	126,0
	badejo	*SPARC ULTRA1* 140	128	143	291,0	109,0

Tabela 6.2: Características dos ambientes de processamento utilizados (estações da *Sun*).

Ambiente/ Rede	Nome da estação	Modelo	Características: *RAM*, *clock*, etc.			
			[*MBytes*]	[*MHz*]	[*MIPS*]	[*MFLOPS*]
FDDI	tubarao	*SPARC20* M50	96	50	134,2	30,5
	piava	*SPARC4 microSPARCII*	88	70	105,4	15,8
	"outras"	*SPARC4 microSPARCII*	88	70	105,4	15,8
Ethernet	tainha	*SPARC10* M10/40	64	40	90,0	10,6
	salmon	*SPARCServer330*	32	25	16,0	2,6
	baiacu	*SPARC2* M4/75	16*	33	28,5	4,2
	"outras"	*SPARC2* M4/75	16**	33	28,5	4,2

* Atualizada recentemente para 46 [*MBytes*].
** Atualizada recentemente para 32 [*MBytes*].

Os sistemas operacionais utilizados nas máquinas relacionadas nas tabelas acima são versões do *UNIX*: *SunOS Release 4.1.3, 4.1.1, SunOS Release 5.5 e 5.5.1* (i.e., versões do sistema *Solaris* na rede *FDDI* e nas estações *SunSPARC ULTRA*, respectivamente).

6.2 Testes e Resultados com os Programas de FPORS Seqüenciais

Nesta seção são apresentados os testes e os resultados obtidos com as implementações seqüenciais de fluxo de potência ótimo com restrições de segurança sem e com remanejamento pós-contingência.

6.2.1 Redes Elétricas Testadas

A Tabela 6.3 apresenta as características dos sistemas elétricos testados referentes a esta seção e à seção 6.3.

Tabela 6.3: Características dos sistemas elétricos testados com os programas de fluxo de potência ótimo com restrições de segurança (FPORS).

Rede	Número de			
	barras	ramos	geradores	contingências
IEEE-30	30	41	6	30
IEEE-118	118	179	54	170
IEEE-300	300	411	69	200
BR-725	725	1212	93	880
BR-810	810	1340	114	200
BR-1663	1663	2349	146	1555

6.2.2 Estudo de Casos

CASO 1: A importância de usar geradores fictícios em implementações de FPORS.

Condições de estudo
1. programa FPORS-I;
2. remanejamento: $\mathbf{\Delta} = \mathbf{0}$
3. carregamento nominal é o que consta do banco de dados ($\mathbf{P}_g^{\text{nom}}$ e $\mathbf{P}_d^{\text{nom}}$);
4. sistemas-testes: BR-725 e *IEEE*-118.

As Tabelas 6.4 e 6.5 mostram o aumento do carregamento dos ramos dos sistemas BR-725 e *IEEE*-118 (através do crescimento da geração e da demanda) e a evolução dos valores de geração fictícia necessários para tornar cada problema viável.

Nestas tabelas (6.4 e 6.5) estão também indicados o número de restrições na base ótima final bem como o número de iterações de dual-simplex revisado[1] (vide Figura 2.3 do Capítulo 2), indicando o aumento do trabalho computacional para resolver o problema à medida que geradores fictícios são necessários.

Tabela 6.4: Evolução da solução do problema de FPORS para o sistema-teste BR-725 em função do carregamento da rede elétrica; inicialização *AC*.

Carregamento		Geração fictícia total $[MW]$	$[MW]$ realocados	Nº restrições na base	Nº it. dsr
Geração	Carga				
$0{,}75\mathbf{P}_g^{nom}$	$0{,}76\mathbf{P}_d^{nom}$	0,00	3289,96	6	7
$0{,}85\mathbf{P}_g^{nom}$	$0{,}86\mathbf{P}_d^{nom}$	49,24	7363,22	30	312
$1{,}00\mathbf{P}_g^{nom}$	$1{,}00\mathbf{P}_d^{nom}$	443,43	9837,66	48	280

Tabela 6.5: Evolução da solução do problema de FPORS para o sistema-teste *IEEE*-118 em função do carregamento da rede elétrica; inicialização *AC*.

Carregamento		Geração fictícia total $[MW]$	$[MW]$ realocados	Nº restrições na base	Nº it. dsr
Geração	Carga				
$1{,}00\mathbf{P}_g^{nom}$	$1{,}00\mathbf{P}_d^{nom}$	0,00	566,44	4	7
$1{,}20\mathbf{P}_g^{nom}$	$1{,}20\mathbf{P}_d^{nom}$	0,00	1050,05	7	10
$1{,}30\mathbf{P}_g^{nom}$	$1{,}30\mathbf{P}_d^{nom}$	18,11	1298,01	9	17

A conclusão extraída destes testes é que, se não fossem empregados geradores fictícios, o problema de FPORS para determinados níveis de carregamento do sistema simplesmente ficaria sem solução. Outra observação é que, quanto mais carregada estiver a rede, maior será o trabalho computacional necessário para resolver o problema e mais restrições serão adicionadas à base do processo PL.

[1] São iterações internas do algoritmo de PL.

CASO 2: Análise comparativa das restrições de segurança tipo Benders e tipo Stott para contingências críticas de um sistema elétrico real.

Condições de estudo
1. programas FPORS-III e FPORS-II;
2. remanejamento: $\Delta \cong \mathbf{0}$
3. carregamento nominal (do banco de dados);
4. inicialização AC;
5. sistema-teste: BR-725;
6. objetivo desvio quadrático;
7. dois pares de contingências críticas são analisadas;
8. cada par possui contingências geograficamente distantes.

O objetivo deste estudo é comparar as restrições de segurança geradas nos métodos Stott e Benders quanto à eficácia para resolver as violações em ramos. O teste foi projetado da seguinte maneira: duas contingências críticas geograficamente distantes são tomadas do sistema Sul-Sudeste (BR-725), de modo que qualquer uma delas quando ocorre individualmente produz sobrecarga.

Dois conjuntos, com duas contingências cada um, foram selecionados para este estudo. A Tabela 6.6 mostra as contingências em análise.

Tabela 6.6: Caracterização das contingências analisadas (BR-725).

Contingência (Simbologia)	Contingência			Sobrecarga
	Ramo nº	Barras (nº/nome)	Localização	Ramo nº
A1	110	182/PREAL–181/PREAL	Sul	81
A2	939	1215/ASSIS–1222/CAPIVARA	Sudeste	966
B1	808	1119/ITABERA–1167/T.PRETO	Sudeste	512
B2	358	550/CHAMINE–551/CHAMINE	Sul	359

Cada conjunto de contingências foi simulado inicialmente como se as contingências ocorressem individualmente e, depois, como contingências múltiplas. Por exemplo, a contingência simples A1 foi processada na implementação FPORS-III; em seguida, a contingência simples A2. Após a solução das contingências A1 e A2 fez-se a simulação da contingência múltipla A1 & A2. O mesmo procedimento foi repetido para o outro conjunto de contingências (B1, B2, B1 & B2). De modo análogo, procedeu-se em relação ao programa FPORS-II.

A Tabela 6.7 resume os resultados obtidos para as situações descritas anteriormente. A Tabela 6.8 mostra os tempos médios de computação das implementações executando em uma estação *SunSPARC4* referentes aos testes da Tabela 6.7; nesta tabela estão incluídos apenas os tempos consumidos pelo processo iterativo do FPORS em cada situação estudada.

Tabela 6.7: Comportamento das implementações de FPORS-III e FPORS-II quanto aos cortes gerados para duas situações de contingências críticas; sistema-teste BR-725.

FPORS	Iterações e restrições	Contingências simples e múltipla					
		A1	A2	A1&A2	B1	B2	B1&B2
Benders (III)	Nº it. caso-base	**2**	**2**	**2**	**2**	**2**	**2**
	Nº it. operação	1	1	1	1	1	1
	Nº it. dsr*	12	20	17	17	13	18
	Nº restrições geradas	**1**	**1**	**1**	**1**	**1**	**1**
	Nº restrições na base	1	1	1	1	1	1
Stott (II)	Nº it. caso-base	2	2	7	3	2	4
	Nº it. operação	1	2	6	3	1	4
	Nº it. dsr	6	6	6	9	6	11
	Nº restrições geradas	1	2	6	3	1	4
	Nº restrições na base	1	1	1	1	1	1

* dsr = dual-simplex revisado (iterações internas; Figura 2.3 do Capítulo 2).

Tabela 6.8: Tempos médios de computação das implementações na estação *SunSPARC4* obtidos nos testes da Tabela 6.7; considerado somente o tempo do processo iterativo do FPORS.

Implementação	Tempos médios de computação [s]					
	A1	A2	A1&A2	B1	B2	B1&B2
FPORS-III	1,40	3,05	**2,49**	1,99	1,48	**2,16**
FPORS-II	0,70	3,33	3,42	4,86	0,46	4,94

Da Tabela 6.7 observa-se que, para a maioria das situações simuladas, o número de restrições geradas pelo método de Benders resultou menor que o número de restrições geradas pelo método de Stott (programa FPORS-II). O mesmo se observa em relação à quantidade de iterações realizadas pelo problema de investimento (i.e., PL do caso-base). Os tempos da implementação FPORS-III são sempre menores que os tempos correspondentes do programa FPORS-II quando são simuladas contingências múltiplas, conforme mostra a Tabela 6.8. Em resumo, os cortes de Benders (i.e., restrições produzidas pelo programa FPORS-III) foram mais eficazes (incisivos) que o cortes gerados a partir da máxima violação, ou seja, segundo a metodologia de Stott.

A explicação para este fato reside na composição da restrição de segurança gerada por cada um dos métodos. Regra geral, a restrição do programa FPORS-III traz mais informações a respeito da contingência que a restrição do programa FPORS-II. Para exemplificar, o exame do conteúdo das restrições mostra o seguinte: a restrição tipo Benders apresenta em média 24% de posições preenchidas com elementos não-nulos (corte denso), enquanto a restrição tipo Stott apresenta apenas 2% de posições preenchidas (corte esparso).

CASO 3: O efeito das variações do vetor Δ sobre os valores da função objetivo, $[MW]$ realocados, nº de iterações e nº de restrições na base final.

Condições de estudo
1. programas FPORS-III, FPORS-II e FPORS-I;
2. remanejamento: Δ de **0** até valores altos;
3. carregamento leve;
4. inicialização *DC*;
5. sistemas-testes: BR-725 e $IEEE$-118;
6. objetivo econômico para o sistema do $IEEE$ e objetivo desvio quadrático para o sistema BR-725;
7. contingências não ordenadas no banco de dados e o programa obtém o *ranking*.

Os valores do vetor Δ referidos nestes experimentos referem-se a uma porcentagem sobre os limites superiores da geração do respectivo banco de dados. Assim, por exemplo, para o sistema-teste BR-725, $\Delta = 8\%$ equivale a $8\%\mathbf{P}_g^{máx}$. As Tabelas 6.9 e 6.10 mostram como varia a função objetivo, o número de iterações e o número de restrições geradas e na base à medida que cresce o vetor Δ para dois sistemas elétricos de características distintas. A Figura 6.4 ilustra a variação da função objetivo MDQ em termos dos valores do vetor Δ para o sistema da Tabela 6.9. Para os testes com os programas FPORS II e III, 0% significa $\cong$ 0%.

Tabela 6.9: Evolução da função objetivo, $[MW]$ realocados, número de iterações, número de restrições geradas e número de restrições na base final com as variações do vetor Δ para as três implementações; sistema-teste BR-725.

FPORS	Valores obtidos	Variações do vetor Δ em [%] de $\mathbf{P}_g^{máx}$					
		0%	4%	7,2%	7,5%	12%	24%
III	Função objetivo (MDQ)*	0,468	0,149	0,051	0,068	0,003	0,000
	$[MW]$ realocados	3483	2316	1360	1322	61	0,00
	Nº it. caso-base	9	25	23	29	2	1
	Nº it. operação	12	35	51	51	7	6
	Nº restrições geradas	8	24	22	28	1	0
	Nº restrições na base	6	6	5	5	1	0
II	Função objetivo (MDQ)	0,442	0,160	0,060	0,064	0,002	0,000
	$[MW]$ realocados	3490	2310	1382	1330	82	0,00
	Nº it. caso-base	9	10	9	10	2	1
	Nº it. operação	9	15	15	18	7	6
	Nº restrições geradas	9	15	15	18	7	6
	Nº restrições na base	6	6	5	5	1	0
I	Função objetivo (MDQ)	0,442	–	–	–	–	–
	$[MW]$ realocados	3490	–	–	–	–	–
	Nº restrições geradas	9	–	–	–	–	–
	Nº restrições na base	6	–	–	–	–	–

* MDQ significa mínimo desvio quadrático.

Tabela 6.10: Evolução da função objetivo, $[MW]$ realocados, número de iterações, número de restrições geradas e número de restrições na base final com as variações do vetor Δ para as três implementações; sistema-teste $IEEE$-118.

FPORS	Valores obtidos	Variações do vetor Δ em [%] de $P_g^{máx}$					
		0%	0,5%	1%	2%	12,5%	15%
III	Função objetivo (E)*	14835,19	14760,40	14704,37	14657,37	14647,29	14647,29
	$[MW]$ realocados	1067	1032	1037	1112	1141	1141
	Nº it. caso-base	7	5	5	5	1	1
	Nº it. operação	12	11	14	24	22	22
	Nº restrições geradas	6	4	4	4	0	0
	Nº restrições na base	4	3	3	3	0	0
II	Função objetivo (E)	14835,33	14818,72	14803,14	14775,81	14648,24	14647,29
	$[MW]$ realocados	1067	1063	1065	1058	1130	1141
	Nº it. caso-base	7	10	6	6	3	1
	Nº it. operação	6	23	13	10	24	22
	Nº restrições geradas	6	23	13	10	24	22
	Nº restrições na base	4	4	3	3	2	0
I	Função objetivo (E)	14835,37	–	–	–	–	–
	$[MW]$ realocados	1067	–	–	–	–	–
	Nº restrições geradas	6	–	–	–	–	–
	Nº restrições na base	4	–	–	–	–	–

* E significa objetivo econômico (custo de operação).

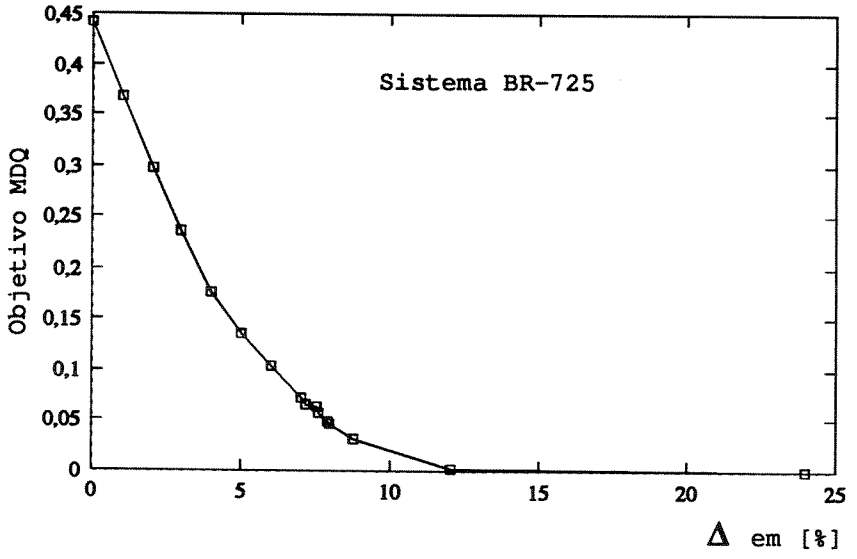


Figura 6.4: Evolução da função objetivo mínimo desvio quadrático para vários valores de Δ; inicialização *DC*.

CASO 4: Comparação dos tempos de computação dos programas de FPORS executando em três modelos diferentes de estações para dois sistemas elétricos.

Condições de estudo
1. programas FPORS-III, FPORS-II e FPORS-I;
2. remanejamento: $\Delta = 0$
3. carregamento leve;
4. sistemas-testes: BR-725 e *IEEE*-300;
5. objetivo desvio quadrático para ambos os sistemas;
6. contingências não ordenadas no banco de dados e o programa obtém o *ranking*.

A Tabela 6.11 mostra os tempos totais médios de computação para os três programas de FPORS, em execução seqüencial, em três máquinas de capacidades diferentes: *SPARC2*, *SPARC4* e *SPARC ULTRA*. Nos testes com os programas FPORS II e III, o remanejamento Δ foi fixado num valor muito próximo de zero (0,001%).

Tabela 6.11: Tempos totais médios de computação consumidos na execução dos programas de FPORS para $\Delta = 0$ em três modelos de estações; inicialização *AC*.

Implementação	Tempos totais médios de computação [*s*]					
	IEEE-300			BR-725		
	SPARC2	*SPARC4*	*SPARC ULTRA*	*SPARC2*	*SPARC4*	*SPARC ULTRA*
FPORS-III	14,45	5,42	2,94	106,19	42,64	20,51
FPORS-II	11,29	4,63	2,20	138,30	66,60	30,04
FPORS-I	10,10	4,48	2,08	88,89	35,48	15,16

Da Tabela 6.11 conclui-se que, para esta condição testada, o programa mais rápido é o FPORS-I (que é uma implementação mais simples e dedicada à situação em que $\Delta = 0$). Para o sistema-teste BR-725, o programa FPORS-III apresentou desempenho bastante próximo ao da implementação FPORS-I; o mesmo não se pode afirmar em relação aos testes com o sistema *IEEE*-300.

Em termos relativos, tomando-se por referência a estação *SPARC2*, as velocidades das máquinas *SPARC4* e *SPARC ULTRA* são aproximadamente 2,2 e 5,4, respectivamente.

CASO 5: Desempenho dos programas FPORS-III e FPORS-II em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para valores distintos de Δ

Condições de estudo
1. programas FPORS-III e FPORS-II;
2. remanejamento: vários valores de Δ;
3. carregamento leve;
4. sistemas-testes: *IEEE*-30, *IEEE*-118 e BR-725;
5. objetivo custo de operação para o sistema *IEEE*-30;
6. objetivo desvio quadrático para os sistemas BR-725 e *IEEE*-118;
7. testes na estação *SunSPARC2*;
8. contingências não ordenadas no banco de dados e o programa obtém o *ranking*.

O objetivo deste estudo de caso é investigar a operacionalidade dos programas FPORS-III e FPORS-II para diversos valores de Δ. As questões básicas que se procura responder com estes testes são: (1) qual é dentre as duas implementações a que apresenta melhor desempenho?; e (2) em quais condições isto ocorre?

A Tabela 6.12 mostra os resultados obtidos para o sistema *IEEE*-30; os tempos de computação envolvidos são muito pequenos por isso não estão mostrados. Pode-se observar que os resultados finais (e.g., valores da função objetivo e restrições na base final) são praticamente coincidentes. Entretanto, para facilitar o entendimento dos resultados que estão apresentados nas tabelas subseqüentes, cabe aqui tecer algumas considerações sobre os resultados apresentados na Tabela 6.12 (vide Figuras 6.2 e 6.3):

- por exemplo, tomando-se $\Delta = 1\%$, o nº de iterações do caso-base é 4 para ambos os programas; ou seja, foram resolvidos 4 PLs do caso-base;
- foram também resolvidos 7 PLs de operação pelo FPORS-II e 6 pelo FPORS-III;
- o programa FPORS-II gerou mais restrições de segurança (7) que o FPORS-III (que gerou apenas 4);
- quanto as restrições que ficaram na base, cada um dos programas finalizou a solução com a mesma quantidade de restrições na base (não está sendo computada neste número a restrição de balanço, $\beta^t \Delta \mathbf{P} = 0$, sempre presente na base);
- o número de iterações dsr refere-se à quantidade de vezes que o algoritmo de PL (cujo diagrama está ilustrado na Figura 2.3) foi iterado; tais iterações são também referidas como iterações internas de PL (e.g., mudanças de base).

Tabela 6.12: Desempenho dos programas FPORS-II e FPORS-III em termos do nº de iterações, nº de restrições geradas e na base final, e valores da função objetivo para diversos valores do vetor Δ; inicialização *DC*; *IEEE*-30.

Valor do vetor Δ	Implementação FPORS-II					
	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Valor da função objetivo [$/h]
	caso-base	operação	caso-base	operação		
0,001%	4	3	3	3	3/3	813,85
1%	**4**	**7**	**3**	**7**	**7/3**	**812,15**
10%	3	10	2	10	10/2	796,31
100%	1	8	0	8	8/0	783,32
200%	1	8	0	8	8/0	783,32
Valor do vetor Δ	Implementação FPORS-III					
	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Valor da função objetivo [$/h]
	caso-base	operação	caso-base	operação		
0,001%	3	5	4	3	3/3	813,84
1%	**4**	**6**	**5**	**7**	**4/3**	**812,12**
10%	14	29	13	20	13/2	797,47
100%	2	13	1	2	0/0	783,36
200%	2	13	1	2	0/0	783,36

A seguir, duas situações são investigadas: valores reduzidos de Δ e valores altos de Δ. Em relação ao sistema mais difícil (BR-725) é analisado também o efeito das contingências múltiplas.

Conforme discutido anteriormente, as magnitudes das componentes do vetor Δ indicam o nível de remanejamento que é permitido aos controles ativos nos estados pós-contingências:

- $\Delta = \mathbf{0}$ significa que nenhum remanejamento é permitido;
- Δ pequeno, por exemplo, 1%, significa que pequena margem de remanejamento é permitida;
- Δ alto, por exemplo, 10% ou 40%, significa que ampla margem de remanejamento é permitida (i.e., o valor da função objetivo diverge pouco do valor correspondente ao primeiro PL do caso-base);
- $\Delta \rightarrow \infty$, todo remanejamento é permitido (i.e., o valor da função objetivo é idêntico ao valor correspondente ao primeiro PL do caso-base).

CASO 5A: Comportamento dos programas FPORS-II e FPORS-III para valores reduzidos de Δ

As Tabelas 6.13 e 6.14 mostram as iterações, as restrições e os tempos totais de computação correspondentes, respectivamente, aos programas FPORS-II e FPORS-III para o sistema-teste *IEEE*-118. Ressalta-se que este banco de dados só apresenta contingências simples.

Tabela 6.13: Desempenho do programa FPORS-II em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para três valores do vetor Δ; inicialização *AC*; *IEEE*-118.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [*s*]
	caso-base	operação	caso-base	operação		
0,001%	6	5	6	5	5/4	8,32
1%	10	20	10	20	20/4	13,85
1,5%	11	20	11	20	20/4	14,37

Tabela 6.14: Desempenho do programa FPORS-III em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para três valores do vetor Δ; inicialização *AC*; *IEEE*-118.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [*s*]
	caso-base	operação	caso-base	operação		
0,001%	6	5	5	10	5/4	7,44
1%	9	9	8	17	8/4	8,42
1,5%	10	11	9	20	9/4	8,65

Duas observações podem ser feitas a partir da análise das Tabelas 6.13 e 6.14: (1) a implementação FPORS-III mostrou-se mais rápida que a FPORS-II; (2) exceto para $\Delta = 0,001\%$, o programa FPORS-II gerou uma maior quantidade de restrições, o que implicou em maior número de iterações de PL e conseqüentemente maior consumo de tempo. Destas observações se conclui que os cortes de Benders (multiplicadores simplex) foram mais eficazes que os de Stott (máxima violação) nos casos apresentados nas tabelas supra-citadas.

Outra conclusão importante, obtida por comparação da situação $\Delta = 0,001\%$ de ambos os programas, é que, uma iteração de operação do FPORS-III mostrou ser mais rápida que uma iteração de operação do FPORS-II (os PLs de operação são executados por rotinas implementadas de maneira ligeiramente diferente). Esta afirmativa é possível tendo em vista que o PL do caso-base é efetuado por rotinas idênticas.

Sistema BR-725: Lista com Contingências Simples.

As Tabelas 6.15 e 6.16 mostram os resultados de simulações com o sistema BR-725 nas duas implementações com vários níveis de remanejamento. Foram escolhidos, deliberadamente,

valores baixos de Δ para realçar as características de cada programa.

Tabela 6.15: Desempenho do programa FPORS-II em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para quatro valores do vetor Δ; inicialização AC; BR-725.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [s]
	caso-base	operação	caso-base	operação		
0,001%	9	9	8	9	9/6	139,13
1%	11	18	10	18	18/6	163,81
2%	11	18	10	18	18/6	165,46
3%	12	21	11	21	21/6	181,22

Tabela 6.16: Desempenho do programa FPORS-III em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para quatro valores do vetor Δ; inicialização AC; BR-725.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [s]
	caso-base	operação	caso-base	operação		
0,001%	9	10	8	22	8/6	105,22
1%	29	45	28	67	28/6	149,15
2%	24	45	23	43	23/6	151,12
3%	19	33	18	46	18/6	139,23

A conclusão extraída destas tabelas para a rede BR-725 com contingências simples na composição da lista é que, os tempos de computação conseguidos com o programa FPORS-III são menores que os obtidos com o FPORS-II, quando a margem de remanejamento permitida é relativamente pequena.

Sistema BR-725: Lista com Contingências Múltiplas Entre os Casos Críticos.

As Tabelas 6.17 e 6.18 mostram os resultados de simulações dos programas FPORS-II e FPORS-III, nas mesmas condições testadas para obter as tabelas acima exceto pelo fato da lista de contingências conter saídas múltiplas de ramos entre os casos críticos (no caso específico, foram consideradas contingências duplas).

Da observação dos resultados das Tabelas 6.17 e 6.18 constata-se acentuadas diferenças nos tempos de computação em favor da implementação FPORS-III (que utiliza cortes por multiplicadores simplex). Estes resultados confirmam as conclusões tiradas a partir do Caso 2.

Tabela 6.17: Desempenho do programa FPORS-II em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para quatro valores do vetor Δ; inicialização *AC*; BR-725.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [*s*]
	caso-base	operação	caso-base	operação		
0,001%	8	9	7	9	9/6	116,69
1%	11	19	10	19	19/6	173,76
2%	12	27	11	27	27/6	205,12
3%	12	21	11	21	21/6	191,87

Tabela 6.18: Desempenho do programa FPORS-III em termos do nº de iterações, nº de restrições na base final e dos tempos de computação para quatro valores do vetor Δ; inicialização *AC*; BR-725.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [*s*]
	caso-base	operação	caso-base	operação		
0,001%	10	13	9	30	9/6	108,62
1%	28	55	27	83	27/6	170,10
2%	19	53	18	86	18/6	161,32
3%	19	37	18	64	18/6	139,57

CASO 5B: Comportamento dos programas FPORS-II e FPORS-III para valores elevados de Δ

Esta seção tem a finalidade de investigar os desempenhos das implementações de FPORS II e III para valores elevados de Δ; inicialização *AC*. As Tabelas 6.19 e 6.20 apresentam os resultados obtidos para o sistema *IEEE*-118; os resultados para o sistema BR-725 estão mostrados nas Tabelas 6.21 e 6.22.

Comparando-se a 5ª coluna das Tabelas 6.19, 6.20, 6.21 e 6.22 verifica-se que o número de iterações internas (dsr) do problema de operação foram maiores para o programa FPORS-III. Isto explica os tempos de computação mais elevados para esta implementação. Chama-se a atenção do leitor para o remanejamento 12% nas Tabelas 6.21 e 6.22, em que as iterações são praticamente iguais e os tempos medidos acompanham esta tendência.

Tabela 6.19: Desempenho do programa FPORS-II para valores elevados de Δ; $IEEE$-118.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [s]
	caso-base	operação	caso-base	operação		
8%	3	9	2	9	9/2	6,96
9%	3	10	2	10	10/2	6,67
10%	3	10	2	10	10/2	7,12
16%	1	16	0	20	16/0	5,87
20%	1	16	0	28	16/0	5,87
40%	1	16	0	28	16/0	5,86

Tabela 6.20: Desempenho do programa FPORS-III para valores elevados de Δ; $IEEE$-118.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [s]
	caso-base	operação	caso-base	operação		
8%	4	17	3	41	3/2	11,95
9%	5	20	4	50	4/2	10,25
10%	3	16	2	40	2/2	10,98
16%	1	16	0	38	0/0	10,97
20%	1	16	0	41	0/0	11,15
40%	1	16	0	56	0/0	11,17

Tabela 6.21: Desempenho do programa FPORS-II para valores elevados de Δ; BR-725.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [s]
	caso-base	operação	caso-base	operação		
8%	7	13	6	13	13/4	122,91
12%	2	7	1	7	7/1	103,50
24%	1	6	0	8	6/0	87,52
32%	1	6	0	6	6/0	88,01

Tabela 6.22: Desempenho do programa FPORS-III para valores elevados de Δ; BR-725.

Valores do vetor Δ	Nº de iterações		Nº de iterações dsr		Nº de restrições geradas/na base	Tempo total [s]
	caso-base	operação	caso-base	operação		
8%	18	33	17	37	17/4	136,64
12%	2	7	1	7	1/1	104,32
24%	1	6	0	22	0/0	103,16
32%	1	6	0	42	0/0	121,96

A explicação para o fato da implementação FPORS-III executar elevado número de iterações internas (o que faz subir o tempo de computação), isto é, muitas mudanças de base, pode ser ilustrada conforme a Figura 6.5.

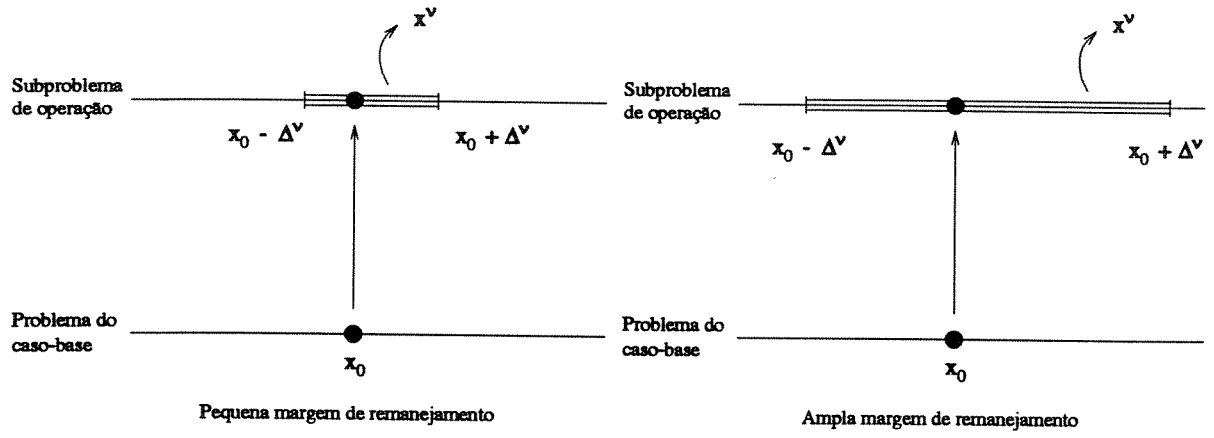


Figura 6.5: Ilustração de uma iteração do método de Benders; quanto mais alto for Δ maior será o espaço de busca do processo de otimização.

A partir deste estudo de caso (Caso 5) as seguintes observações gerais podem ser explicitadas:

- para faixas pequenas de remanejamento, a implementação FPORS-III apresenta melhor *performance* que a implementação FPORS-II;
- para faixas grandes de remanejamento, a situação se inverte, a implementação FPORS-II é mais vantajosa que a FPORS-III;
- estas constatações confirmam os resultados obtidos no Caso 2; justamente quando os cortes são mais necessários para mudar a solução do caso-base (i.e., quando Δ é pequeno), o programa cuja geração do corte se baseia nos multiplicadores simplex apresenta desempenho superior;
- quanto ao FPORS-III, a cada subproblema de operação não necessariamente é gerado um novo corte. Por exemplo, para Δ estreito (pequeno ou próximo de zero) é mais provável a necessidade de geração de mais cortes, embora a solução possa evoluir mais rapidamente para a solução final;
- para valores muito altos de Δ (e.g., as situações em que o número de restrições na base é nula) ambos os programas realizam os PLs de operação apenas para verificar se a contingência é remanejável ou não. Isto sugere uma interessante idéia no sentido de se criar uma heurística para determinar *a priori* se um dado Δ pode ser considerado alto ou baixo mediante uma lista de contingências e um sistema-teste;
- à medida que a margem de remanejamento cresce, os tempos de computação para ambos os programas tendem a se estabilizar, ou seja, não crescem indefinidamente;
- na implementação FPORS-II, o PL de operação é sempre utilizado para incluir uma restrição de segurança (ou seja, pela entrada 2 da Figura 2.3) ao contrário do FPORS-III.

6.3 Estudo de Casos com os Programas de FPORS Paralelizados Segundo o Modelo Produtor-Consumidor

Nos testes desta seção, os bancos de dados são aqueles indicados na Tabela 6.3; quando alguma variante nos dados desta tabela for necessária ela será oportunamente referida no estudo de caso correspondente.

6.3.1 Estudo de Casos

CASO 6: Tempos médios das partes componentes das implementações paralelas de FPORS nas redes *Ethernet* e *FDDI*.

Condições de estudo
1. programas FPORS-I, FPORS-II e FPORS-III;
2. carregamento leve;
3. inicialização *AC*;
4. sistemas-testes: BR-725 e BR-1663;
5. tamanho da lista de contingências indicado na Tabela 6.3;
6. redes *Ethernet* e *FDDI* livres de usuários.

Nas Tabelas 6.23 a 6.26, as medidas indicadas foram efetuadas nas seguintes partes dos programas: inicialização do *PVM3*, *I/O*, cálculo do ponto inicial e difusão dos dados, cálculo do FPO caso-base e análise de contingências para obtenção do *ranking*. Os tempos de duração destas componentes não dependem do tipo de programa que está executando; seja FPORS-I, FPORS-II ou FPORS-III estas partes são comuns às três implementações.

O principal objetivo deste estudo de caso é mostrar a ordem de grandeza das partes componentes dos programas e a evolução dos tempos de processamento, inicialização e comunicação à medida que se aumenta o número de estações. A seguir, essas componentes são discutidas.

(1.) Tempo de inicialização da aplicação *PVM3*:

Num Centro de Controle, a parcela de tempo correspondente à inicialização do *PVM3* deve ser negligenciada após ter sido partido o programa. Em outras palavras, uma vez iniciada a execução de uma função de análise de rede elétrica, a aplicação deverá ficar permanentemente processando. Das tabelas pode-se observar que, este tempo cresce com o aumento do número de estações alocadas para o processamento distribuído, e independe do sistema sob análise.

(2.) Tempo de *I/O*:

Nos testes efetuados, observa-se que, esta parcela depende da dimensão do banco de dados do sistema elétrico e dos arquivos de resultados, contudo, numa situação real, a função de *I/O* é normalmente substituída por uma leitura/escrita direta em uma memória virtual (que é muito rápida), ou seja, sem acesso a disco rígido. Isto significa que esta parcela na prática poderá ser negligenciada.

(3.) Tempo de obtenção do ponto inicial e difusão dos dados:

Esta componente do tempo cresce com o número de máquinas configuradas já que nela está incluída o tempo de difusão de dados e de resultados parciais aos demais processos. É uma parcela importante no cálculo do desempenho dos sistemas de programação/processamento distribuído.

(4.) Tempo para calcular o FPO caso-base:

É o tempo consumido na execução dos cálculos do PL do caso-base; nesta etapa, todas as restrições de contingências são relaxadas e apenas a rede intacta é viabilizada através do ajuste dos controles do sistema. É uma parcela fixa mas depende do sistema sob análise, do seu carregamento (carga/geração/fluxos) e do ambiente de processamento utilizado. Deve ser considerada no cálculo do desempenho dos sistemas de programação/processamento distribuído.

Tabela 6.23: Tempos médios das partes componentes dos programas FPORS-I, FPORS-II e FPORS-III em segundos; ambiente *Ethernet*; sistema BR-725.

Nº de estações	Tempo médios das partes componentes [*s*]				
	Inic.*PVM*	*I/O*	Inic. *AC* & difusão	Caso-base	*Ranking*
1	0,01	5,40	1,80	0,71	35,43
2	0,60	5,40	1,91	0,71	17,91
3	0,75	5,40	2,16	0,71	12,14
4	0,87	5,40	2,24	0,71	9,13
5	0,90	5,40	2,93	0,71	7,37
6	0,93	5,40	3,30	0,71	6,26
7	1,12	5,40	3,33	0,71	5,41
8	1,13	5,40	3,40	0,71	4,81

(5.) Tempo para obter o *ranking* das contingências:

Este tempo inclui a análise de contingências (neste caso, *screening* $1P\theta$), a transmissão dos índices calculados, a ordenação da lista a partir dos índices de severidade (neste caso, PI_{MW}), e também o *broadcasting* da lista ordenada a todos os processos. É uma parcela que decresce quase linearmente com o aumento do número de máquinas. É sem dúvida considerada no cálculo do desempenho.

Tabela 6.24: Tempos médios das partes componentes dos programas FPORS-I, FPORS-II e FPORS-III em segundos; ambiente *FDDI*; sistema BR-725.

Nº de estações	Tempo médios das partes componentes [s]				
	Inic.*PVM*	*I/O*	Inic. *AC* & difusão	Caso-base	*Ranking*
1	0,01	1,86	0,82	0,58	13,68
2	0,40	1,86	0,99	0,58	7,11
3	0,74	1,86	1,08	0,58	9,85
4	0,91	1,86	1,12	0,58	8,33
5	0,94	1,86	1,21	0,58	6,82
6	0,94	1,86	1,35	0,58	6,29
7	0,98	1,86	1,37	0,58	6,13
8	0,99	1,86	1,84	0,58	5,19

As medidas que se referem exclusivamente ao processo iterativo do FPORS não estão indicadas aqui pelas seguintes razões: (a) são partes distintas nos três programas tanto em desempenho e quanto aos cálculos que são realizados; e (b) dependem do remanejamento Δ permitido.

Com base na exposição feita acima, os ganhos em tempos de execução e outras medidas de desempenho que são mostrados nesta seção (por exemplo, *speedup*, eficiência, eficácia e tempos médios de computação) são tomados considerando-se sempre as parcelas de tempos indicadas na expressão seguinte:

Tempo para p *hosts* = (3.) + (4.) + (5.) + Tempo do processo iterativo do FPORS

O tempo correspondente à execução em um *host* ($p = 1$), e referente às parcelas acima indicadas, é neste trabalho designado por tempo-base.

Tabela 6.25: Tempos médios das partes componentes dos programas FPORS-I, FPORS-II e FPORS-III em segundos; ambiente *Ethernet*; sistema BR-1663.

Nº de estações	Tempo médios das partes componentes [s]				
	Inic.*PVM*	*I/O*	Inic. *AC* & difusão	Caso-base	*Ranking*
1	0,01	11,40	7,54	2,01	109,88
2	0,60	11,40	9,02	2,01	55,72
3	0,75	11,40	9,32	2,01	37,04
4	0,87	11,40	10,09	2,01	28,42
5	0,90	11,40	10,10	2,01	22,64
6	0,93	11,40	10,30	2,01	19,08
7	1,12	11,40	10,50	2,01	16,68
8	1,13	11,40	10,60	2,01	15,12

Tabela 6.26: Tempos médios das partes componentes dos programas FPORS-I, FPORS-II e FPORS-III em segundos; ambiente *FDDI*; sistema BR-1663.

Nº de estações	Tempo médios das partes componentes [s]				
	Inic.*PVM*	*I/O*	Inic. *AC* & difusão	Caso-base	*Ranking*
1	0,01	4,43	3,77	1,79	41,27
2	0,40	4,43	4,17	1,79	21,73
3	0,74	4,43	4,21	1,79	14,60
4	0,91	4,43	4,40	1,79	10,99
5	0,94	4,43	4,57	1,79	9,13
6	0,94	4,43	4,79	1,79	7,42
7	0,98	4,43	5,00	1,79	6,72
8	0,99	4,43	6,44	1,79	6,61

CASO 7: Avaliação do desempenho da implementação paralela na rede *Ethernet* e a influência do tamanho da lista: *speedup* (S), eficiência (E) e eficácia (η).

Condições de estudo
1. programa FPORS-I;
2. remanejamento: $\Delta = 0$;
3. carregamento leve;
4. inicialização *AC*;
5. sistemas-testes: BR-725 e BR-1663;
6. rede *Ethernet* livre de usuários;
7. contingências não ordenadas no banco de dados e o programa obtém o *ranking*.

Dois tamanhos de listas de contingências são consideradas: (1) longa (880 contingências para o sistema BR-725, e 1555 para o BR-1663); e (2) curta (100 contingências para o sistema BR-725). A Tabela 6.27 mostra os tempos de referência utilizados no cálculo do desempenho paralelo neste estudo de caso.

Tabela 6.27: Tempos-base em segundos do programa FPORS-I para os sistemas testados; ambiente *Ethernet*.

Rede	Lista	Nº de contingências	Tempo-base [s]
BR-725	longa	880	82,40
BR-725	curta	100	13,49
BR-1663	longa	1555	105,91

A Figura 6.6 ilustra as curvas do *speedup* para o sistema BR-725 com dois tamanhos de listas; as Figuras 6.7 e 6.8 ilustram as correspondentes eficiência e eficácia.

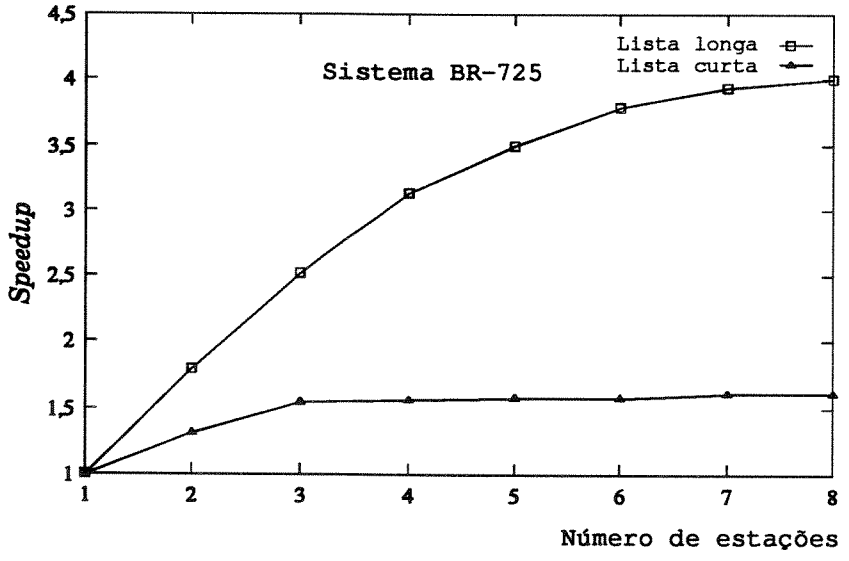


Figura 6.6: Curvas do *speedup* (S) para dois tamanhos de listas de contingências.

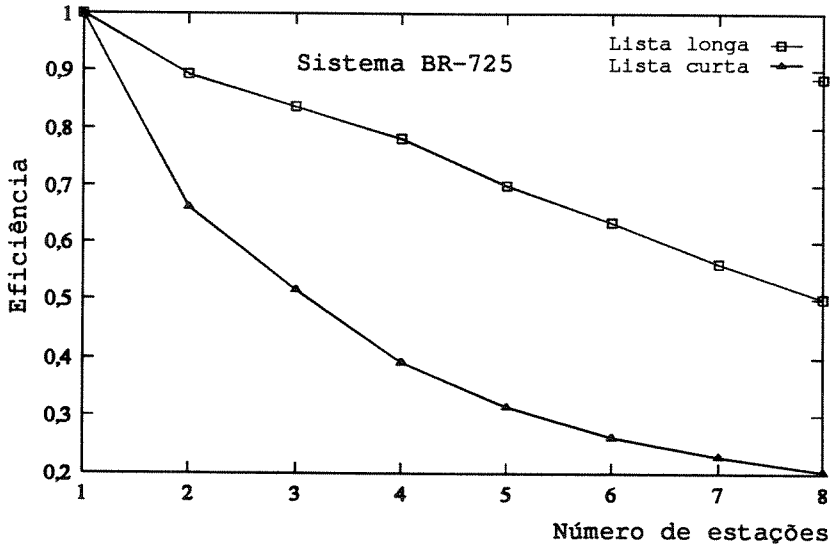


Figura 6.7: Curvas da eficiência (E) para dois tamanhos de listas de contingências.

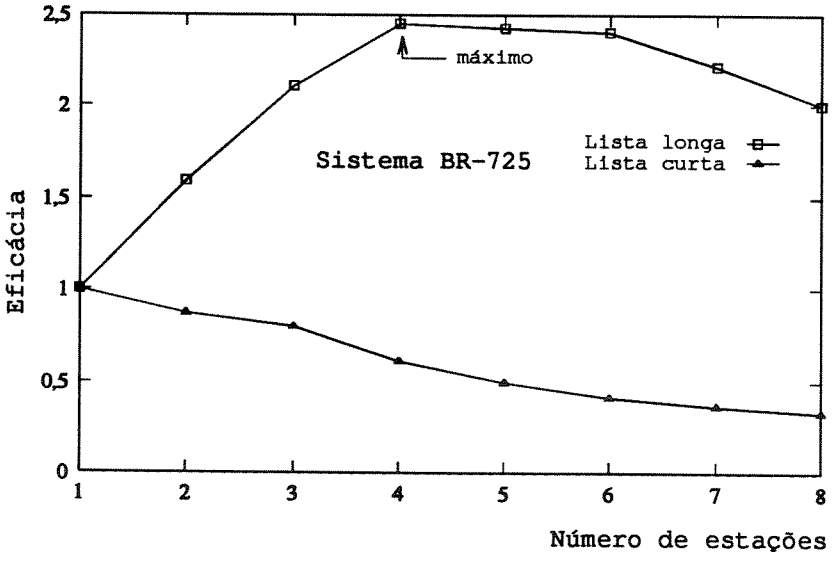


Figura 6.8: Curvas da eficácia (η) para dois tamanhos de listas de contingências.

A Tabela 6.28 mostra como varia o número de cortes gerados para os dois sistemas da Figura 6.9 à medida que se aumenta o número de máquinas para o processamento distribuído do FPORS. O número de restrições de segurança ativas na solução final em cada sistema testado é 6 e 5, respectivamente, para as redes BR-725 e BR-1663.

Tabela 6.28: Evolução do número de cortes efetivamente gerados com o aumento do número de máquinas.

Nº de estações	Nº de cortes gerados	
	BR-725	BR-1663
1	7	6
2	11	8
3	15	10
4	16	10
5	19	13
6	25	13
7	25	13
8	25	13

A Figura 6.9 compara o desempenho paralelo para dois sistemas brasileiros com listas longas de contingências.

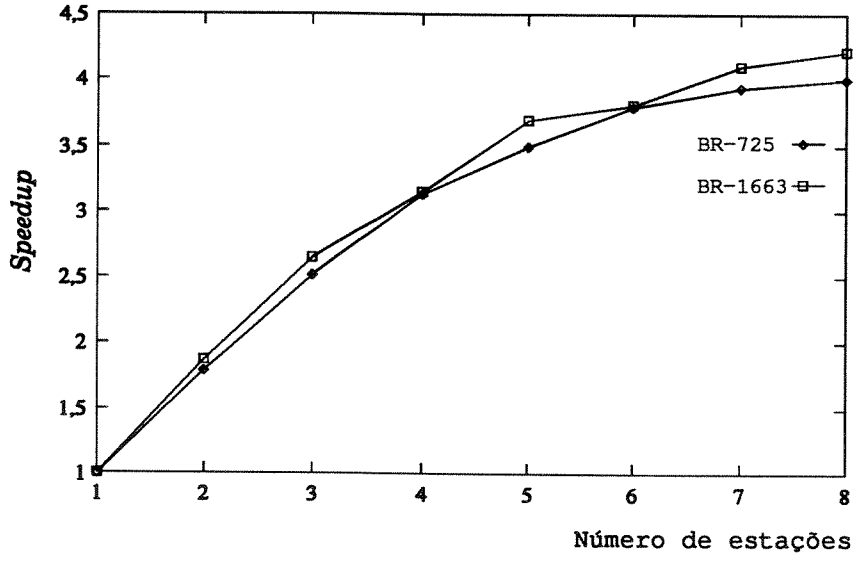


Figura 6.9: Curvas do *speedup* (S) para sistemas elétricos de dimensões diferentes, com listas longas de contingências.

CASO 8: Avaliação do desempenho da implementação paralela na rede *Ethernet* para dois sistemas elétricos diferentes com listas de contingências de tamanhos médios: *speedup* (S), eficiência (E) e eficácia (η).

Condições de estudo
1. programa FPORS-I;
2. remanejamento: $\Delta = 0$;
3. carregamento leve;
4. inicialização *AC*;
5. sistemas-testes: *IEEE*-300 e BR-810;
6. 4 estações da rede *Ethernet* livre de usuários;
7. listas com 200 contingências;
8. contingências não ordenadas no banco de dados e o programa obtém o *ranking*.

A Tabela 6.29 mostra o desempenho do programa FPORS-I na rede *Ethernet* para sistemas elétricos cujas listas de contingências são de dimensões médias. Observa-se que os melhores ganhos ocorrem para 2 estações para ambos os sistemas.

Tabela 6.29: Medidas de desempenho paralelo do programa FPORS-I para dois sistemas elétricos de tamanhos diferentes; testes feitos com 1, 2, 3 e 4 estações da rede *Ethernet*.

Nº de	*IEEE*-300			BR-810		
estações	*Speedup*	Eficiência	Eficácia	*Speedup*	Eficiência	Eficácia
1	1,00	1,00	1,00	1,00	1,00	1,00
2	1,54	**0,77**	**1,19**	1,67	**0,84**	1,40
3	1,76	0,59	1,04	2,20	0,73	**1,61**
4	1,90	0,48	0,91	2,52	0,63	1,59

Este teste é importante uma vez que listas de contingências de tamanhos médios são mais comuns na prática dos Centros de Controle.

CASO 9: Efeito da ordenação da lista de contingências sobre os desempenhos da implementação paralela nas redes *Ethernet* e *FDDI*.

Condições de estudo
1. programa FPORS-I;
2. remanejamento: $\Delta = 0$;
3. carregamento leve;
4. inicialização *AC*;
5. sistema-teste: BR-725;
6. 5 estações das redes *Ethernet* e *FDDI* livres de usuários;
7. 3 situações da lista de contingências são estudadas.

Neste teste, o objetivo é verificar o papel da ordenação das contingências no desempenho da implementação paralela. Três situações foram simuladas:

(A) A lista está não ordenada no banco de dados e o programa faz a classificação das contingências antes de passar ao cálculo iterativo do FPORS; este é um caso no qual as contingências críticas não são conhecidas; o programa consome um certo tempo formando o *ranking*; tempo-base de 82,40 [*s*] na rede *Ethernet*; tempo-base de 33,61 [*s*] na rede *FDDI*;

(B) A lista está ordenada no banco de dados e o programa não precisa fazer a classificação das contingências antes de passar ao cálculo iterativo do FPORS; este é um caso no qual as contingências críticas são conhecidas *a priori*; o programa economiza tempo de processamento; é uma situação que se aproxima bastante da prática de um Centro de Controle; tempo-base de 46,97 [*s*] na rede *Ethernet*; tempo-base de 19,93 [*s*] na rede *FDDI*;

(C) A lista está não ordenada no banco de dados e não se faz a classificação das contingências antes de passar ao cálculo iterativo do FPORS; este é um caso no qual nada se sabe sobre as contingências críticas; o programa tende a consumir um tempo excessivo no FPORS; tempo-base de 131,04 [*s*] na rede *Ethernet*; tempo-base de 52,88 [*s*] na rede *FDDI*.

As Figuras 6.10 e 6.11 ilustram os desempenhos para as três situações mencionadas para ambos os ambientes de processamento.

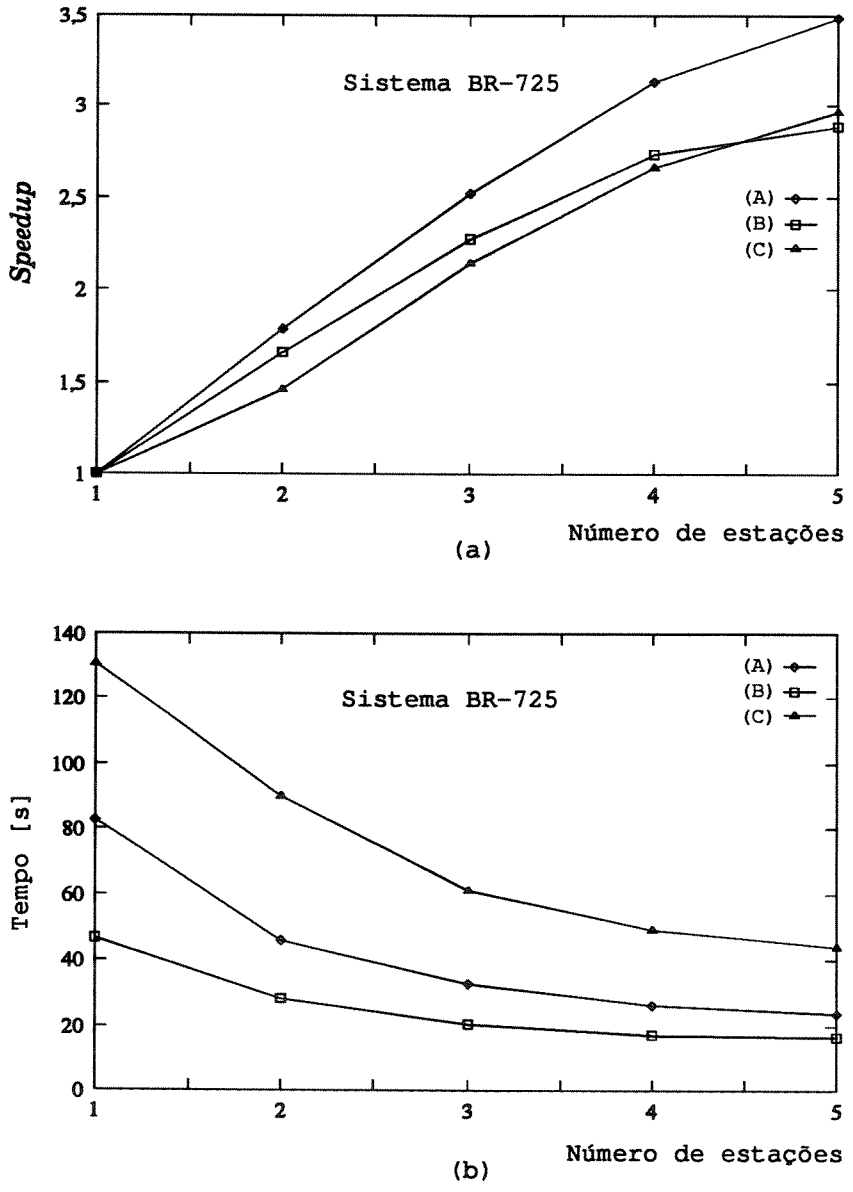


Figura 6.10: Medidas de desempenho paralelo na rede *Ethernet*: (a) *speedup*; (b) tempos médios de computação em segundos.

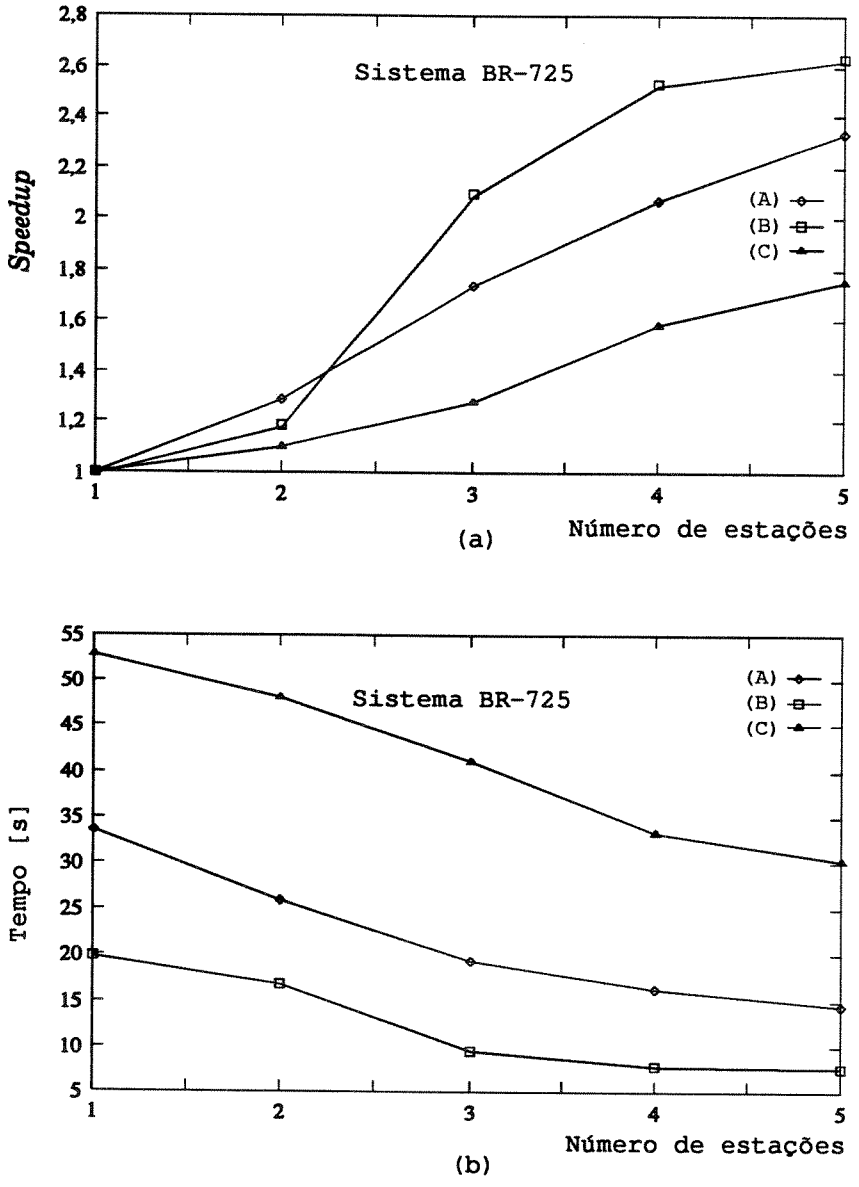


Figura 6.11: Medidas de desempenho paralelo na rede *FDDI*: (a) *speedup*; (b) tempos médios de computação em segundos.

CASO 10: Influência da capacidade de remanejamento sobre o desempenho das implementações paralelas.

Condições de estudo
1. programas FPORS-II e FPORS-III;
2. diversos valores de remanejamento permitido;
3. carregamento leve;
4. inicialização *AC*;
5. sistema-teste: BR-725;
6. 6 estações da rede *Ethernet* livre de usuários;
7. contingências não ordenadas no banco de dados e o programa obtém o *ranking*.

As Figuras 6.12 e 6.13 ilustram as curvas do *speedup* (S) para os programas FPORS-II e FPORS-III, cobrindo quatro valores distintos de remanejamento permitido.

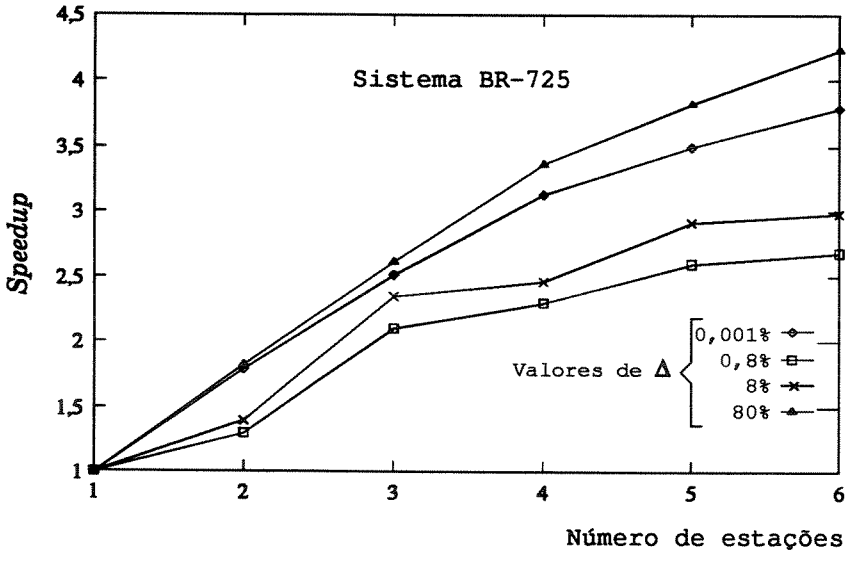


Figura 6.12: Curvas do *speedup* (S) para o programa FPORS-II para vários valores de Δ na rede *Ethernet*.

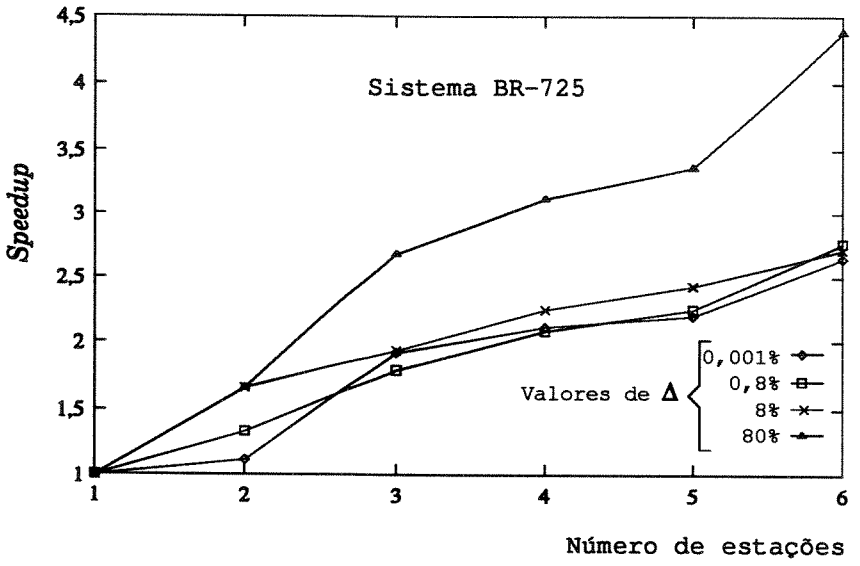


Figura 6.13: Curvas do *speedup* (S) para o programa FPORS-III para vários valores de Δ na rede *Ethernet*.

Observa-se das Figuras 6.12 e 6.13 que os programas apresentam desempenhos muito próximos (ou similares) para 2 e 3 estações na máquina paralela virtual. Para o nível mais alto de remanejamento testado, 80%, contata-se uma tendência de crescimento dos *speedups* à medida que aumenta o número de estações.

6.4 Testes e Resultados com o Programa ranking Paralelizado Segundo os Modelos *Pipeline* e *Hosted*.

Esta seção apresenta os testes e os resultados obtidos com o programa **ranking**. Diversos testes são efetuados, alguns com as implementações distribuídas operando com uma base de dados apenas, como se faz tradicionalmente, e outros com um fluxo de dez bases de dados, conforme a proposta apresentada no Capítulo 5.

6.4.1 Redes Elétricas Testadas

A Tabela 6.30 apresenta as características dos sistemas elétricos testados nesta seção.

Tabela 6.30: Características dos sistemas elétricos testados nas comparações dos modelos *Hosted* e *Pipeline*; *IEEE*-300, BR-810 e BR-725 possuem 69, 114 e 93 geradores, respectivamente.

Rede	Número de			*Screening* de	
	barras	ramos	contingências	sobrecarga	correção
IEEE-300 (A)	300	411	300	300	50
IEEE-300 (B)	300	411	200	200	50
BR-810 (A)	810	1340	300	300	50
BR-810 (B)	810	1340	100	100	15
BR-725	725	1212	880	880	880

6.4.2 Estudo de Casos

CASO 11: A influência dos modos de comunicação síncrono e assíncrono no desempenho do programa paralelo **ranking**.

Condições de estudo
1. modelo *Hosted* submetido a uma base de dados;
2. rede *FDDI* livre com 6 estações;
3. sistema-teste: BR-810(A).

Na seção 4.5.1 foram definidos os modos de comunicação de casos de contingências aos processadores: (1) síncrono, que corresponde à forma trivial de paralelização de uma lista de casos independentes (N_c/p); (2) assíncrono 1, que se refere ao envio pelo processo *master* de um caso por vez ao processo solicitante, imediatamente à chegada do sinal de conclusão da tarefa; e (3) o assíncrono 2, que é uma variante do modo assíncrono 1, permite o envio de mais de um caso por solicitação. Designando-se por S_{env} o número de casos de contingências distribuídos aos processos por vez, neste teste, o modo assíncrono 2 foi simulado com S_{env} igual a 6; os modos síncrono e assíncrono 1 têm seus desempenhos comparados com o assíncrono 2.

A Figura 6.14 ilustra os tempos médios de computação, incluindo-se inicialização do *PVM3*, *I/O* e comunicação, para as três modalidades descritas.

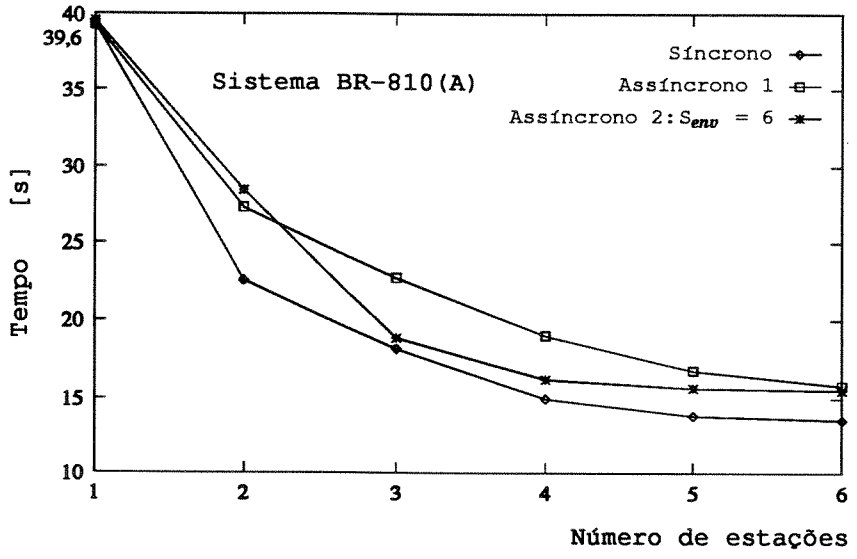


Figura 6.14: Tempos para a rede BR-810(A) com o programa paralelo ranking; modelo *Hosted* com uma base de dados; três modos de comunicação; assíncrono 2 com "envelope" de tamanho igual a 6.

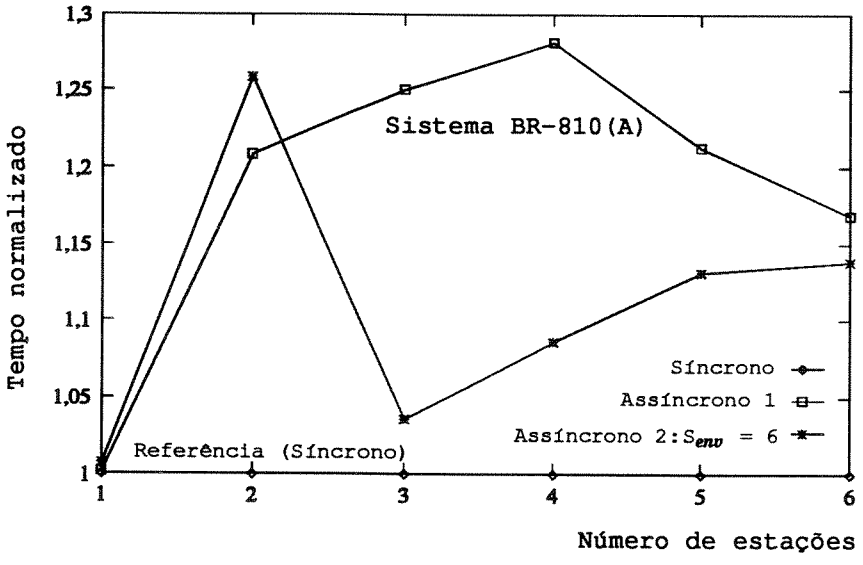


Figura 6.15: Tempos normalizados pelos correspondentes valores do modo síncrono; (obtidos a partir dos valores reais em segundos da Figura 6.14).

Da observação das Figuras 6.14 e 6.15 constata-se que, o modo síncrono resulta em geral nos menores tempos médios de computação; o modo assíncrono 1 apresenta o desempenho mais desfavorável dentre os pesquisados. O modo assíncrono 2, com valor de S_{env} igual a 6, mostra desempenho intermediário entre as modalidades síncrona e assíncrona 1.

O modo síncrono é o que menos utiliza o meio de comunicação durante a execução da análise das contingências. Cada processo "sabe" *a priori* quais contingências deverá calcular e somente ao finalizar suas tarefas comunica os resultados ao *master*. O modo assíncrono 1, por sua vez, solicita um novo caso de contingência a cada caso concluído; dentre os modos assíncronos testados é o que utiliza o meio de comunicação com mais freqüência.

Em linhas gerais, com base nestes resultados, conclui-se que, quanto mais se utiliza a rede de comunicação (que trabalha de forma seqüencial) pior é o desempenho neste tipo de aplicação. Ressalta-se, entretanto, que na análise de contingências o desbalanço de carga computacional entre tarefas é em geral pouco acentuado. Todavia, podem ocorrer situações nas quais o *overhead* extra de comunicação devido ao assincronismo seja desprezível em comparação com o desbalanço de carga e resultar em melhores desempenhos para o modo assíncrono.

CASO 12: Demonstração da viabilidade de classificar as contingências de uma lista a partir do índice PI_{MDQ}, obtido do critério de correção de sobrecarga através de solução de PL.

Condições de estudo
1. sistemas-testes: $IEEE$-300(B) e BR-725;
2. é indiferente se é utilizada a versão seqüencial ou as versões distribuídas do programa **ranking**.

Nestes testes, o objetivo consiste em verificar a eficácia do índice PI_{MDQ} para distinguir os casos mais graves de contingências (i.e., os casos críticos), bem como a relação desse índice com o tradicional PI_{MW} para sobrecargas.

A rede BR-725 é uma rede elétrica do Sistema Interligado Brasileiro com uma lista de 880 casos de contingência simples e múltipla. É importante o teste com este sistema para mostrar o efeito da nova classificação sobre um sistema real de grande porte, onde contingências consideradas difíceis estão presentes. A rede $IEEE$-300(B) é um banco de dados padrão do $IEEE$ com uma lista de 200 casos de contingência simples e múltipla.

As Tabelas 6.31 e 6.32 mostram os resumos dos arquivos de saída emitidos pelo programa **ranking**. Nestas tabelas estão mostradas as duas classificações ((1) sobrecarga, PI_{MW}; e (2) correção, PI_{MDQ}) para os casos de contingências que figuram no topo de cada lista, portanto

as contingências mais críticas. O restante dos casos foi omitido porque as classificações são completamente coincidentes.

Tabela 6.31: Classificação das contingências para o sistema BR-725 resultantes do programa **ranking** segundo as sobrecargas (PI_{MW}) e segundo as correções das sobrecargas (PI_{MDQ}).

Ordens de severidade		Contingência		Índices		Desvio total	Conceito
sobrecarga	correção	ramo	terminais				
		Nº	NI NF	PI_{MW}	PI_{MDQ}	[MW]	R/F
1ª	2ª	808	1119–1167	41,9	0,133	2472,1	R
2ª	4ª	939	1215–1222	29,0	0,044	364,3	R
3ª	3ª	110	182–181	4,3	0,051	49,8	R
4ª	1ª	208	372–830	2,7	0,135	856,2	R
5ª	5ª	358	550–551	1,2	0,002	2,0	R
6ª	6ª	461	803–704	1,1	0,001	7,4	R
7ª	7ª	736	1087–1103	1,0	0,000	1,2	R
8ª	8ª	993	1248–1249	0,0	0,000	0,0	R

As Tabelas 6.31 e 6.32 comparam as duas classificações mostrando os valores dos índices PI_{MW} e PI_{MDQ} para cada contingência e o correspondente desvio total absoluto de geração em [MW] ([MW] realocados). Nestas tabelas, os significados de R e F são: R significa que não foi preciso usar geradores fictícios na solução; F, em caso contrário.

Tabela 6.32: Classificação das contingências para o sistema $IEEE$-300(B) resultantes do programa **ranking** segundo as sobrecargas (PI_{MW}) e segundo as correções das sobrecargas (PI_{MDQ}).

Ordens de severidade		Contingência		Índices		Desvio total	Conceito
sobrecarga	correção	ramo	terminais				
		Nº	NI NF	PI_{MW}	PI_{MDQ}	[MW]	R/F
1ª	1ª	181	119–120	24,0	22,667	4498,8	F
2ª	2ª	350	63–64	3,3	0,058	192,2	R
3ª	3ª	116	62–64	3,3	0,058	192,2	R
4ª	4ª	231	146–147	1,4	0,001	67,4	R
5ª	5ª	10	9006–9007	0,0	0,000	0,0	R

Observa-se das tabelas mostradas acima que a coerência entre as duas classificações é visível. Isto sugere que a classificação baseada no índice PI_{MDQ} pode ser usada como uma alternativa ao *screening* por sobrecargas ou como uma composição dessas duas classificações. Todavia, quanto à rapidez na obtenção do *ranking*, o método PI_{MDQ} é em geral mais custoso computacionalmente.

CASO 13: Avaliação do desempenho da implementação paralela do programa **ranking** nas redes *FDDI* e *Ethernet*: tempos médios de computação, *speedup* (S), eficiência (E) e eficácia (η).

Condições de estudo
1. modelo *Hosted* submetido a uma base de dados;
2. sistemas-testes: BR-810(A) e *IEEE*-300(A);
3. redes *FDDI* e *Ethernet* livres de usuários;
4. modo síncrono.

As Tabelas 6.33 e 6.34 mostram os tempos médios de computação em segundos nas diferentes partes do programa **ranking** executando na rede *FDDI* com uma base de dados, para os sistemas-testes BR-810(A) e *IEEE*-300(A). As medidas usuais de desempenho paralelo estão também indicadas nestas tabelas, tomando-se por base o tempo de computação para 1 *host* (excluindo-se inicialização do *PVM*3 e *I/O*) (reporte-se ao Caso 6, seção 6.3 para mais detalhes).

As Tabelas 6.35 e 6.36 mostram os tempos médios de computação em segundos nas diferentes partes do programa **ranking** executando na rede *Ethernet* com uma base de dados, para os sistemas-testes BR-810(A) e *IEEE*-300(A).

Tabela 6.33: Tempos de computação do programa **ranking** em segundos; uma base de dados; medidas feitas nas seguintes partes do programa: (a) inicialização do *PVM*3; (b) *I/O*; (c) cálculo do ponto inicial e difusão dos dados; (d) cálculo do FPO caso-base; (e) *ranking* de sobrecargas; (f) *ranking* de correção de sobrecargas. Sistema-teste BR-810(A); ambiente *FDDI*; inicialização *DC*.

Número de estações	Tempos médios de computação [s]							Desempenhos		
	(a)	(b)	(c)	(d)	(e)	(f)	Tempo total	S	E	η
1	0,01	1,62	0,54	1,38	5,84	29,86	39,25	1,00	1,00	1,00
2	0,38	1,62	0,72	1,38	3,04	15,41	22,55	1,83	**0,92**	1,68
3	0,47	1,62	0,76	1,38	2,35	11,62	18,20	2,34	0,78	1,83
4	0,54	1,62	0,85	1,38	1,69	8,80	14,88	2,96	0,74	**2,19**
5	0,59	1,62	0,93	1,38	1,38	11,20	17,10	2,53	0,51	1,29
6	0,58	1,62	0,99	1,38	1,21	7,78	13,56	3,31	0,55	1,82
7	0,65	1,62	1,09	1,38	1,17	6,46	12,37	3,72	0,53	1,97
8	0,80	1,62	1,16	1,38	1,09	5,92	11,97	**3,94**	0,49	1,93

Tabela 6.34: Tempos de computação do programa **ranking** em segundos; uma base de dados; medidas feitas nas seguintes partes do programa: (a) inicialização do *PVM3*; (b) *I/O*; (c) cálculo do ponto inicial e difusão dos dados; (d) cálculo do FPO caso-base; (e) *ranking* de sobrecargas; (f) *ranking* de correção de sobrecargas. Sistema-teste *IEEE*-300(A); ambiente *FDDI*; inicialização *DC*.

Número de estações	Tempos médios de computação [s]							Desempenhos		
	(a)	(b)	(c)	(d)	(e)	(f)	Tempo total	S	E	η
1	0,01	0,83	0,15	0,30	1,96	1,86	5,11	1,00	1,00	1,00
2	0,37	0,83	0,22	0,30	1,12	0,96	3,80	1,64	**0,82**	1,34
3	0,41	0,83	0,25	0,30	0,78	0,68	3,25	2,12	0,71	**1,51**
4	0,62	0,83	0,29	0,30	0,62	0,56	3,22	2,41	0,60	1,45
5	0,55	0,83	0,31	0,30	0,56	0,45	3,00	2,64	0,53	1,40
6	0,59	0,83	0,35	0,30	0,46	0,42	2,95	2,79	0,47	1,31
7	0,66	0,83	0,37	0,30	0,45	0,41	3,02	2,79	0,40	1,12
8	0,78	0,83	0,41	0,30	0,40	0,37	3,09	**2,89**	0,36	1,04

Tabela 6.35: Tempos de computação do programa **ranking** em segundos; uma base de dados; medidas feitas nas seguintes partes do programa: (a) inicialização do *PVM3*; (b) *I/O*; (c) cálculo do ponto inicial e difusão dos dados; (d) cálculo do FPO caso-base; (e) *ranking* de sobrecargas; (f) *ranking* de correção de sobrecargas. Sistema-teste BR-810(A); ambiente *Ethernet*; inicialização *DC*.

Número de estações	Tempos médios de computação [s]							Desempenho		
	(a)	(b)	(c)	(d)	(e)	(f)	Tempo total	S	E	η
1	0,01	4,08	1,01	2,01	14,08	66,41	87,60	1,00	1,00	1,00
2	0,38	4,08	1,28	2,01	7,60	34,23	49,58	1,85	**0,93**	1,72
3	0,39	4,08	1,40	2,01	5,34	25,50	38,72	2,44	0,81	1,98
4	0,51	4,08	1,55	2,01	4,10	19,45	31,70	3,08	0,77	**2,37**
5	0,51	4,08	1,73	2,01	3,48	23,77	35,58	2,69	0,54	1,45
6	0,69	4,08	1,93	2,01	3,14	16,90	28,75	3,48	0,58	2,02
7	0,69	4,08	2,13	2,01	2,96	13,90	25,77	3,98	0,57	2,27
8	1,20	4,08	2,27	2,01	2,88	12,86	25,30	4,17	0,52	2,17
9	1,23	4,08	2,41	2,01	2,80	12,82	25,35	4,17	0,46	1,92
10	1,31	4,08	2,42	2,01	2,80	11,72	24,34	**4,41**	0,44	1,94
11	1,30	4,08	2,59	2,01	4,16	11,80	25,94	4,06	0,37	1,50

Tabela 6.36: Tempos de computação do programa **ranking** em segundos; uma base de dados; medidas feitas nas seguintes partes do programa: (a) inicialização do $PVM3$; (b) I/O; (c) cálculo do ponto inicial e difusão dos dados; (d) cálculo do FPO caso-base; (e) *ranking* de sobrecargas; (f) *ranking* de correção de sobrecargas. Sistema-teste $IEEE$-300(A); ambiente *Ethernet*; inicialização *DC*.

Número de estações	Tempos médios de computação [s]							Desempenho		
	(a)	(b)	(c)	(d)	(e)	(f)	Tempo total	S	E	η
1	0,01	1,81	0,29	0,42	4,31	4,14	10,98	1,00	1,00	1,00
2	0,34	1,81	0,41	0,42	2,45	2,12	7,55	1,70	**0,85**	1,45
3	0,37	1,81	0,44	0,42	1,70	1,49	6,23	2,26	0,75	**1,70**
4	0,48	1,81	0,51	0,42	1,41	1,20	5,83	2,59	0,65	1,68
5	0,49	1,81	0,57	0,42	1,21	0,96	5,46	2,90	0,58	1,68
6	0,63	1,81	0,64	0,42	1,12	0,89	5,51	2,98	0,50	1,49
7	0,62	1,81	0,67	0,42	1,12	0,83	5,47	3,01	0,43	1,29
8	0,94	1,81	0,70	0,42	1,05	0,79	5,71	3,09	0,39	1,21
9	1,14	1,81	0,70	0,42	1,06	0,68	5,81	3,20	0,36	1,15
10	1,17	1,81	0,70	0,42	1,06	0,64	5,80	**3,25**	0,33	1,07
11	1,17	1,81	0,70	0,42	1,28	0,64	6,02	3,01	0,27	0,81

Nas Tabelas 6.33, 6.34, 6.35 e 6.36, mostradas acima, os melhores desempenhos estão realçados em negrito. Observa-se que, em geral, os mais altos *speedups* não correspondem às melhores eficiências. A coluna **(c)** de cada tabela fornece os tempos de cálculo do ponto inicial e difusão dos dados; pode-se observar como estes valores crescem à medida que aumenta o número de processadores alocados. De maneira quase similar, os tempos da coluna **(a)** evoluem com o crescimento do número de processadores. Por outro lado, os tempos das partes efetivamente paralelizadas (vide colunas **(e)** e **(f)**) decrescem com o aumento do número de processadores. Comparando-se os tempos de computação obtidos na rede *FDDI* com os tempos de computação obtidos na rede *Ethernet* verifica-se que as estações do ambiente *FDDI* são aproximadamente 2,2 vezes mais rápidas.

Um aspecto digno de nota é que, se fossem considerados apenas os tempos das partes efetivamente paralelizadas (vide colunas **(e)** e **(f)**), os ganhos com a paralelização seriam realmente muito bons. Contudo, acredita-se que este procedimento não é o indicado num estudo realista.

CASO 14: O efeito do uso compartilhado do ambiente de processamento distribuído no desempenho do programa **ranking.**

Condições de estudo
1. modelo *Hosted* submetido a uma base de dados;
2. sistemas-testes: BR-810(A) e *IEEE*-300(A);
3. rede *FDDI* em duas condições distintas: livre e ocupada;
4. modo síncrono.

As Figuras 6.16 e 6.17 ilustram o comportamento dos tempos médios de computação em duas condições distintas de utilização: rede livre e rede moderadamente ocupada. Os testes foram efetuados durante um dia normal de trabalho no laboratório com usuários compartilhando as mesmas máquinas empregadas, inclusive através de processamento distribuído.

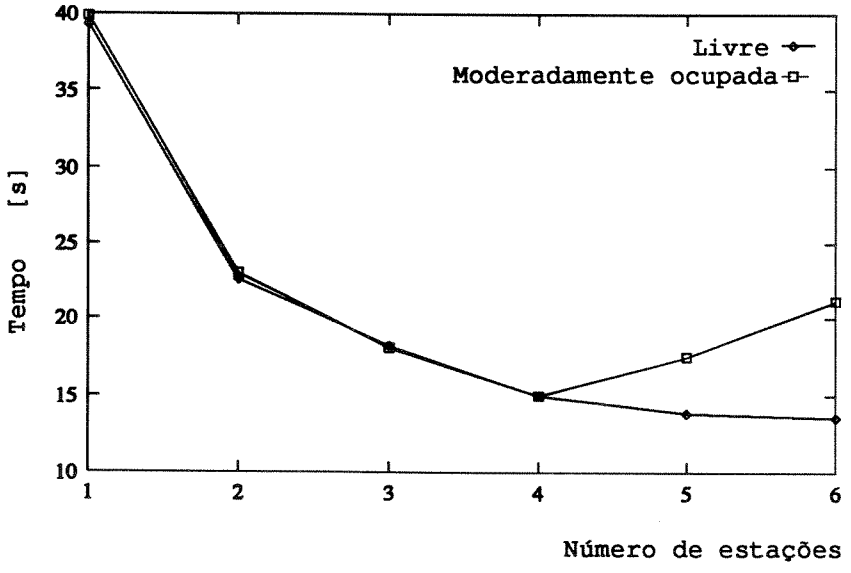


Figura 6.16: Tempos para a rede BR-810(A) com o programa paralelo ranking com a rede *FDDI* nas condições livre e ocupada; modelo *Hosted* com uma base de dados.

O efeito do uso compartilhado mostrou-se mais acentuado durante a execução do sistema-teste *IEEE*-300(A), cujos tempos de computação são substancialmente menores que os tempos de computação do sistema BR-810(A). A conclusão é que, o uso compartilhado, dependendo do grau de utilização dos recursos no instante da execução, pode atrasar consideravelmente o processamento de uma aplicação.

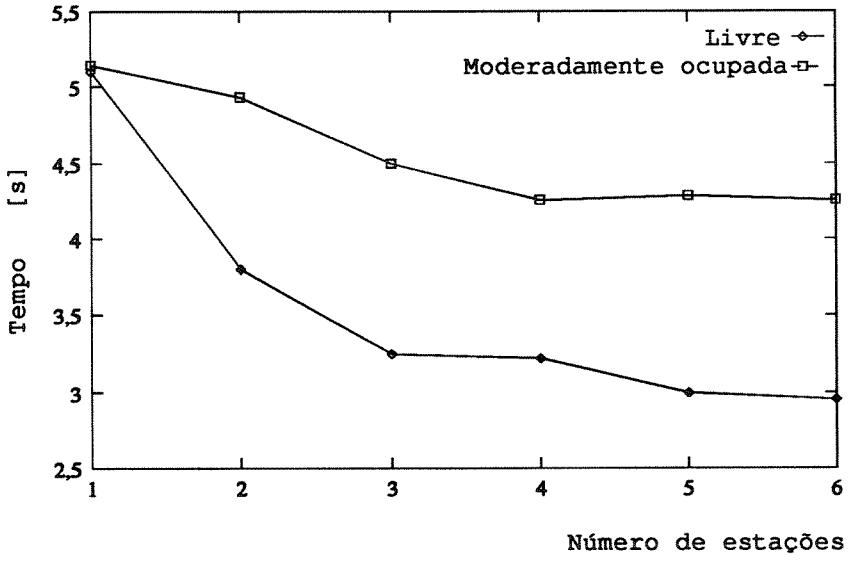


Figura 6.17: Tempos para a rede *IEEE*-300(A) com o programa paralelo ranking com a rede *FDDI* nas condições livre e ocupada; modelo *Hosted* com uma base de dados.

CASO 15: Avaliação comparativa dos desempenhos dos modelos *Hosted* e *Pipeline* com um fluxo de dados na rede *FDDI.*

Condições de estudo
1. programa **ranking** paralelizado conforme os modelos *Hosted* e *Pipeline* submetidos a dez bases de dados;
2. sistemas-testes: *IEEE*-300(A) e (B), BR-810(A) e (B)
3. sistemas-testes com listas curtas e sistemas-testes com listas longas;
4. testes na rede *FDDI* com até 8 estações livres.

Rede *IEEE*-300 com uma Lista Longa de Contingências (*IEEE*-300(A)).

Resultados obtidos com o sistema *IEEE*-300 com uma lista de 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas são dados na Figura 6.18. Os tempos de computação obtidos com o modelo *Pipeline* situam-se em torno de 75% dos tempos de computação do modelo *Hosted* para máquinas paralelas virtuais com 5 e 6 estações, embora para três e quatro estações o *Hosted* apresenta melhor *performance.*

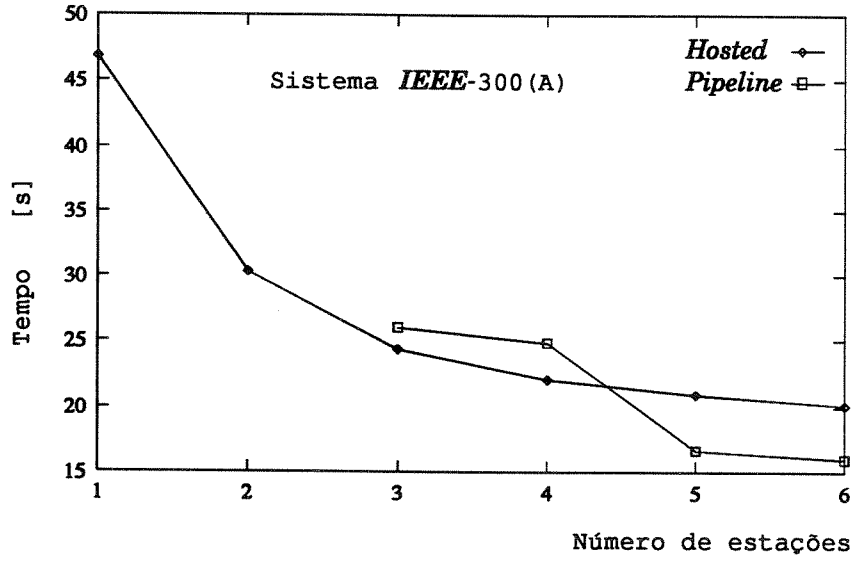


Figura 6.18: Tempos para a rede *IEEE*-300(A) com 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas.

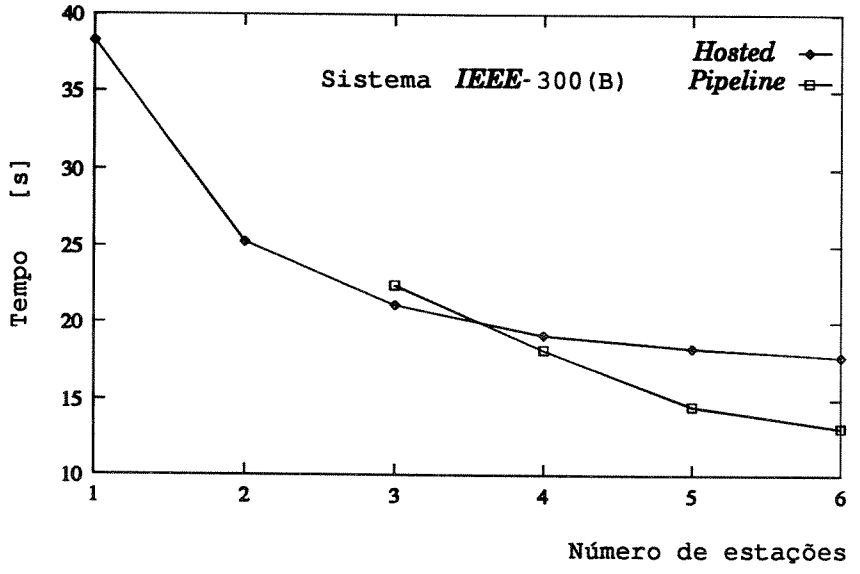


Figura 6.19: Tempos para a rede *IEEE*-300(B) com 200 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas.

Rede *IEEE*-300 com uma Lista Curta de Contingências (*IEEE*-300(B)).

Resultados obtidos com o sistema *IEEE*-300 com uma lista de 200 contingências na obtenção

do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas são dados na Figura 6.19. Os tempos de computação obtidos com o modelo *Pipeline* são significativamente melhores (i.e., o *Pipeline* é mais rápido) que os tempos de computação obtidos com o modelo *Hosted* para máquinas paralelas virtuais a partir de 4 estações. Para três estações, os modelos *Pipeline* e *Hosted* apresentam tempos muito próximos, mas o *Hosted* é melhor.

Rede do Sul-Sudeste do Brasil com uma Lista Longa de Contingências (BR-810(A)).

Resultados obtidos com o sistema Sul-Sudeste do Brasil de 810 barras com uma lista de 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas são dados na Figura 6.20. Os tempos de computação do modelo *Hosted* são significativamente melhores para máquinas virtuais com 3 e 4 estações, a exemplo do que foi observado para o sistema *IEEE*-300(A) (vide Figura 6.18). À medida que cresce o número de estações o modelo *Pipeline* torna-se competitivo com o *Hosted*, chegando a coincidir os tempos de computação para 7 e 8 estações. A razão para o desempenho relativamente bom do *Hosted* neste caso é o grande número de contingências; isto faz com que os tempos consumidos nos estágios seqüenciais do *Pipeline* (estágios A e D) tornem diluídos se comparados com os tempos gastos nos estágios paralelos (estágios B e C). Pode-se supor que, para uma grande lista de contingências, um elevado número de estações deve estar disponível para que superior desempenho do *Pipeline* seja atingido. Isto reporta ao conceito de escalabilidade, aspecto no qual o *Pipeline* é em geral superior ao *Hosted*, uma vez que nele a saturação, embora também presente, sofre considerável atraso.

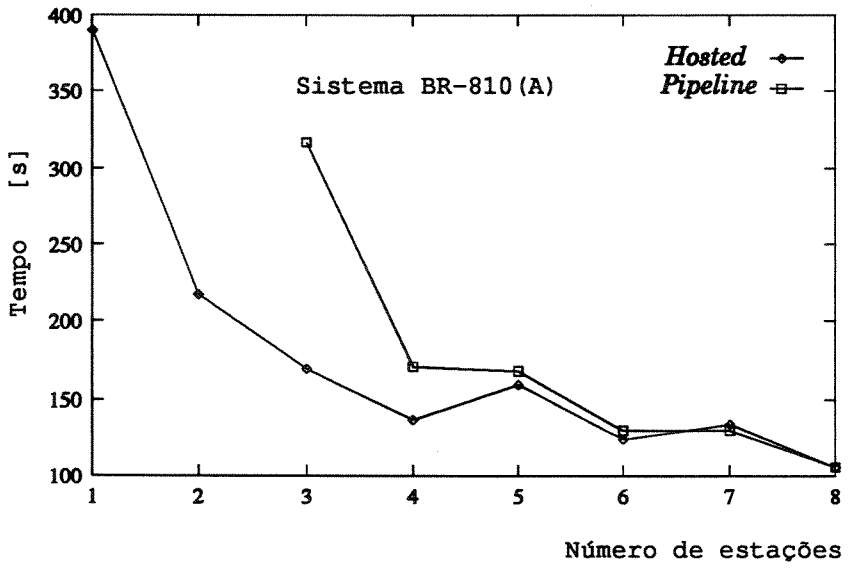


Figura 6.20: Tempos para a rede BR-810(A) com 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas.

Rede do Sul-Sudeste do Brasil com uma Lista Curta de Contingências (BR-810(B)).

Resultados obtidos com o sistema Sul-Sudeste do Brasil de 810 barras com uma lista de 100 contingências na obtenção do *ranking* de sobrecargas e 15 contingências no cálculo da correção de sobrecargas são dados na Figura 6.21. Como na Figura 6.19, o *Pipeline* apresenta menores tempos de computação para máquinas paralelas virtuais a partir de 4 estações. Este resultado confirma novamente a habilidade do *Pipeline* de fornecer bons desempenhos até mesmo quando a granularidade não é tão grande quanto aquelas requeridas pelo modelo *Hosted*.

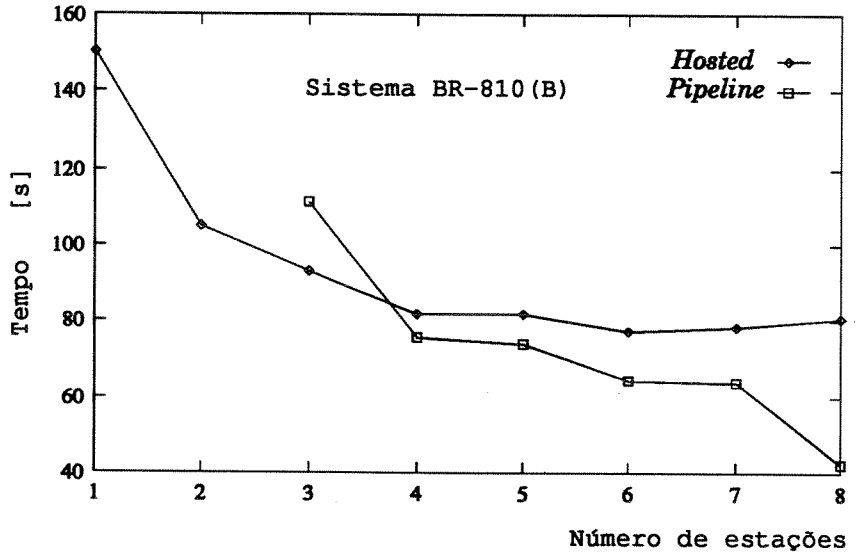


Figura 6.21: Tempos para a rede BR-810(B) com 100 contingências na obtenção do *ranking* de sobrecargas e 15 contingências no cálculo da correção de sobrecargas.

Comparação dos Desempenhos.

A Figura 6.22 ilustra os desempenhos relativos dos modelos *Pipeline* e *Hosted*. Para cada situação, definida por um par formado por um dado número de máquinas e um sistema-teste, os tempos normalizados são dados pela relação do tempo de computação obtido com o modelo *Hosted* e o correspondente tempo de computação obtido com o *Pipeline*. A figura claramente mostra que o modelo *Pipeline* proposto tende a resultar tempos mais curtos quando o número de máquinas (estações) aumenta; o efeito prático final é um desejável atraso da saturação dos ganhos de paralelização à medida que cresce o número de processadores alocados.

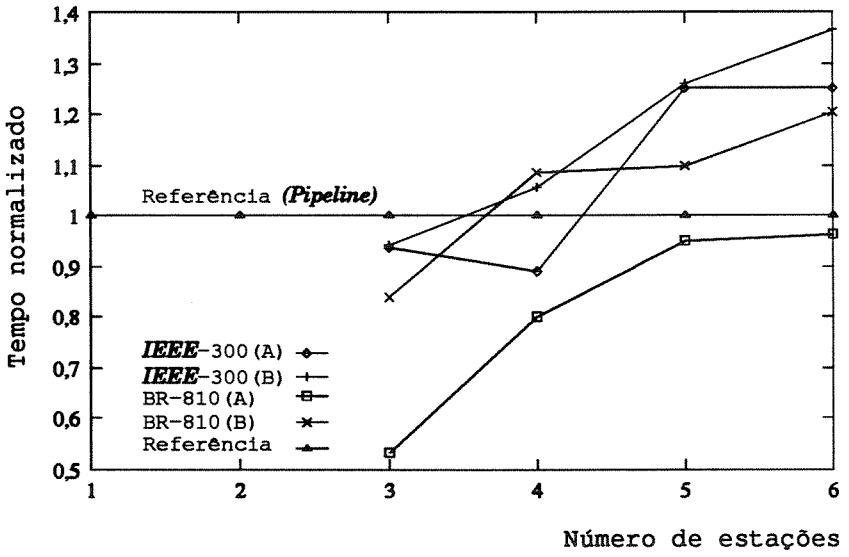


Figura 6.22: Tempos normalizados para o modelo *Hosted* tomando os correspondentes tempos obtidos com o modelo *Pipeline* como referência.

CASO 16: O efeito da falha de uma estação em tempo de execução sobre o desempenho do modelo *Pipeline*.

Condições de estudo
1. programa **ranking** paralelizado conforme os modelos *Hosted* e *Pipeline* submetidos a dez bases de dados;
2. simulada a falha de uma das máquinas;
3. sistemas-testes: *IEEE*-300(A) e BR-810(A);
4. testes na rede *FDDI* com 5 estações livres.

Neste teste foi definida uma configuração inicial de estações, conforme mostra a Tabela 6.37, e durante a execução de forma aleatória foi simulada a falha de um *host* (através de PVMFDELHOST()), no presente caso, a estação de nome pintado; a medição dos tempos foi repetida para 5 simulações. Nas medidas realizadas, o instante da ocorrência da falha variou entre a execução da 5ª e da 6ª base de dados.

Observou-se neste teste que, durante a falha, a máquina remanescente do estágio afetado (no caso, o estágio B) assumiu a carga computacional que era de responsabilidade da estação pintado, ou seja, por causa da falha nenhuma contingência deixou de ser calculada. Após a entrada do *host* substituto, os casos de contingências foram reassumidos; e, alguns segundos depois, o processamento foi concluído sem qualquer prejuízo em relação aos resultados finais. Um justificável atraso no tempo total de computação é o preço pago pela confiabilidade conseguida com a característica dinâmica da aplicação, conforme mostra a Tabela 6.38. Os tempos da

coluna "Estático" correspondem aos respectivos valores dos gráficos ilustrados nas Figuras 6.18 e 6.20 para 5 estações.

Tabela 6.37: Máquinas na configuração: antes e depois da falha.

Máquinas na configuração	
Antes da falha	Pós-falha
piava	piava
cascudo	cascudo
pintado*	tuvira**
dourado	dourado
lambari	lambari

* Máquina em falha.
** Máquina substituta.

Tabela 6.38: Comportamento do modelo *Pipeline* sob a falha de uma máquina.

Rede	Tempo total de computação [s]				Acréscimos [%]	
	Estático		Dinâmico			
	síncrono	assíncrono	síncrono	assíncrono	síncrono	assíncrono
IEEE-300(A)	16,68	25,22	23,47	27,24	40,71	8,01
BR-810(A)	167,89	203,85	178,53	203,90	6,40	0,02

CASO 17: O efeito da inicialização *AC* (cálculo do ponto inicial) sobre os desempenhos dos modelos *Pipeline* e *Hosted* com um fluxo de dados.

Condições de estudo
1. programa **ranking** paralelizado conforme os modelos *Hosted* e *Pipeline* submetidos a dez bases de dados;
2. sistemas-testes: *IEEE*-300(A) e BR-810(A);
3. comparações entre as inicializações *DC* e *AC*;
4. testes na rede *FDDI* com 8 estações livres.

Rede *IEEE*-300(A).

Resultados dos tempos de computação com inicializações *DC* e *AC* para os modelos *Pipeline* e *Hosted* estão ilustrados na Figura 6.23. Seja com inicialização *DC* ou *AC*, o *Pipeline* apresenta desempenho melhor que o modelo *Hosted*. Comparando-se *DC* e *AC*, observa-se que, para este sistema-teste, os tempos de computação nos correspondentes modelos são bem próximos.

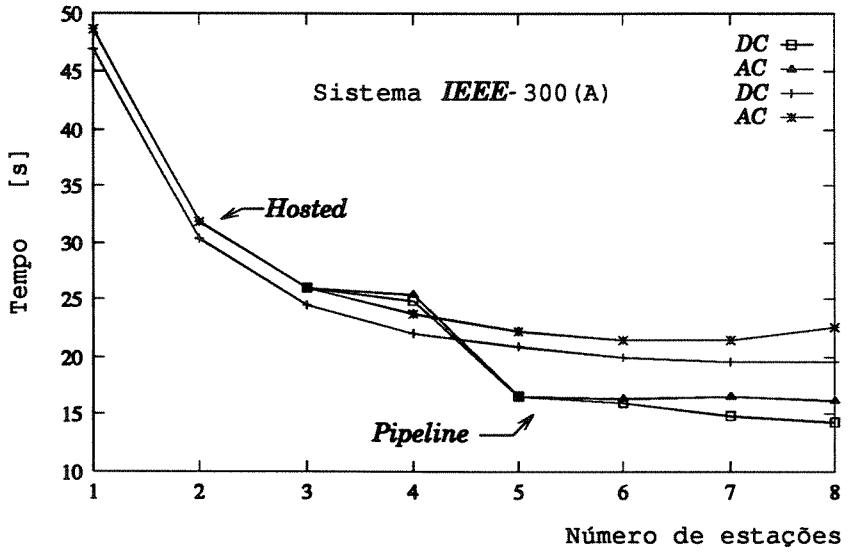


Figura 6.23: Tempos para a rede *IEEE*-300(A) com 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas: comparações entre as inicializações *DC* e *AC*.

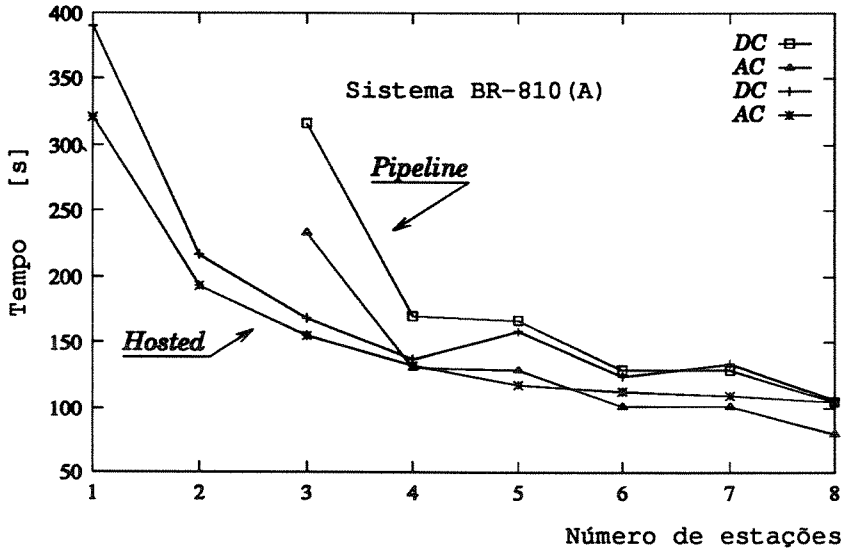


Figura 6.24: Tempos para a rede BR-810(A) com 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas: comparações entre as inicializações *DC* e *AC*.

Rede do Sul-Sudeste do Brasil de 810 Barras (BR-810(A)).

Resultados dos tempos de computação com inicializações *DC* e *AC* para os modelos *Pipeline*

e *Hosted* estão ilustrados na Figura 6.24. O comportamento dos modelos *Pipeline* e *Hosted* é similar ao mostrado na Figura 6.20, ou seja, independente da inicialização DC ou AC. Um aspecto digno de nota é que, os tempos de computação com a inicialização DC são maiores que os tempos de computação quando se usa a inicialização AC (referindo-se inclusive aos tempos do processo PL e de análise de contingências). Isto pode ser explicado da seguinte maneira. Observando-se o número de iterações dsr (i.e., iterações internas de PL), constata-se que, na inicialização DC, as contingências ao serem processadas na obtenção do *ranking* PI_{MDQ} consomem uma quantidade maior de iterações que no caso AC; o que se traduz num acréscimo do tempo de computação.

CASO 18: Avaliação comparativa dos desempenhos dos modelos *Hosted* e *Pipeline* com um fluxo de dados na rede *Ethernet*.

Condições de estudo
1. programa **ranking** paralelizado conforme os modelos *Hosted* e *Pipeline* submetidos a dez bases de dados;
2. sistema-teste: $IEEE$-300(A);
3. sistema-teste com lista longa;
4. testes na rede *Ethernet* com 6 estações livres.

A Figura 6.25 ilustra o comportamento dos modelos *Hosted* e *Pipeline* na rede *Ethernet* para o sistema-teste $IEEE$-300(A).

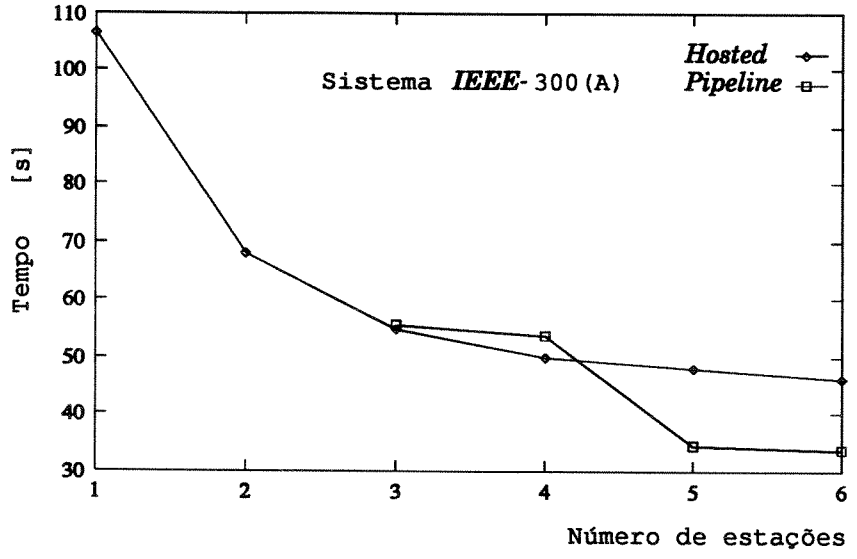


Figura 6.25: Tempos para a rede $IEEE$-300(A) com 300 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas (rede *Ethernet*).

Os resultados deste estudo de caso confirmam o desempenho superior do *Pipeline* em relação ao *Hosted* a partir de 5 estações, conforme foi verificado anteriormente na Figura 6.18 (referente ao outro ambiente de processamento).

CASO 19: Avaliação comparativa dos desempenhos dos modelos *Hosted* e *Pipeline* no modo assíncrono com um fluxo de dados na rede *FDDI.*

Condições de estudo
1. programa **ranking** paralelizado conforme os modelos *Hosted* e *Pipeline* submetidos a dez bases de dados;
2. sistemas-testes: *IEEE*-300(B) e BR-810(B);
3. modo assíncrono;
4. testes na rede *FDDI* com 6 estações livres.

As Figuras 6.26 e 6.27 ilustram as variações dos tempos de computação com o número de estações para dois sistemas-testes, para ambos os modelos *Pipeline* e *Hosted*, onde as contingências são distribuídas segundo o modo assíncrono 1. O modelo *Pipeline* ainda assim apresenta melhor desempenho que o *Hosted.*

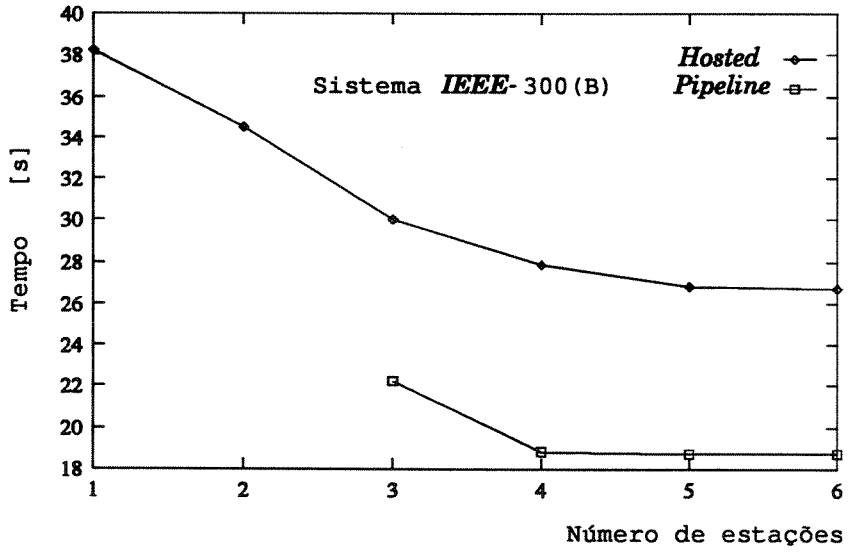


Figura 6.26: Tempos para a rede *IEEE*-300(B) com 200 contingências na obtenção do *ranking* de sobrecargas e 50 contingências no cálculo da correção de sobrecargas; modo assíncrono.

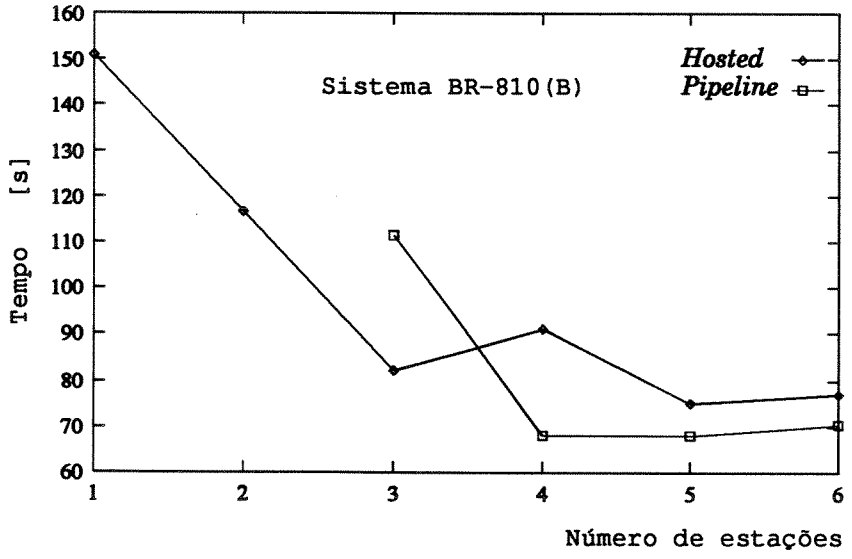


Figura 6.27: Tempos para a rede BR-810(B) com 100 contingências na obtenção do *ranking* de sobrecargas e 15 contingências no cálculo da correção de sobrecargas; modo assíncrono.

Capítulo 7

CONCLUSÕES

Este trabalho apresentou um estudo de funções de análise de segurança estática de sistemas de energia elétrica; especificamente foram abordados os problemas de análise de contingências e o cálculo do fluxo de potência ótimo com restrições de segurança. Modernos ambientes de processamento distribuído foram empregados, providos com os meios *Ethernet* e fibras ópticas.

Na análise de contingências foram estudados e implementados métodos diretos e métodos iterativos. Os métodos diretos, baseados em compensação e atualização de fatores triangulares e já bastante otimizados, foram comparados aos métodos iterativos. Melhoramentos foram introduzidos nos métodos iterativos do gradiente conjugado pré-condicionado (como a ordenação *minimum spanning tree* e um pré-condicionador adequado ao problema de análise de contingências) para torná-los competitivos com os métodos diretos; contingências simples e múltiplas são analisadas.

O fluxo de potência ótimo com restrições de segurança foi abordado através de linearizações do modelo da rede elétrica, o que permitiu o tratamento deste problema de computação intensiva via modelagem para programação linear. O algoritmo de PL utilizado é o dual-simplex revisado proposto em [12]. Técnicas de relaxação de restrições e análise de contingências para construir restrições lineares de segurança foram utilizadas, como uma extensão dos desenvolvimentos feitos em [23] e [43]. A inclusão de capacidades corretivas pós-contingências foi implementada por meio de duas abordagens: (1) extensão do algoritmo de fluxo de potência ótimo com restrições de segurança estudado em [43] para incorporar restrições de acoplamento entre as variáveis do estado básico e pós-contingência; esta implementação utiliza relaxação e construção da restrição a partir da máxima violação (são os chamados cortes esparsos); (2) implementação que usa a técnica de decomposição de Benders e se baseia na construção da restrição a partir do vetor de multiplicadores simplex correspondentes à solução de subproblemas de operação (são os chamados cortes densos).

As implementações de fluxo de potência ótimo com restrições de segurança e de análise de contingências foram transportadas para ambientes de processamento distribuído. O sistema *PVM3* foi utilizado como ferramenta de programação distribuída. Estações de trabalho interligadas em rede formando uma *LAN* foram utilizadas como ambiente de execução das aplicações concorrentes.

Contribuições

Diante do exposto, as principais contribuições deste trabalho podem ser resumidas como a seguir:

- Demonstração da viabilidade de utilização de sistemas de processamento distribuído na análise de segurança de sistemas elétricos nos modernos Centros de Supervisão e Controle;
- Elaboração de rotinas de análise de contingências confiáveis e baseadas nas mais modernas técnicas de programação com esparsidade de matrizes e vetores;
- Melhoramentos nos métodos iterativos para análise de contingências e cálculos de fluxo de potência, que os tornam competitivos com os otimizados métodos diretos, mesmo para sistemas de pequeno porte e computadores convencionais;
- Elaboração de implementações de fluxo de potência ótimo com restrições de segurança (i.e., alívio de violações em ramos por meio do ajuste de controles ativos do sistema) robustas e confiáveis, que incorporam geradores fictícios, contingências simples e múltiplas, capacidade de remanejamento pós-contingência e multi-segmentação da função objetivo não-linear;
- Elaboração de programas distribuídos a partir do sistema *PVM3*, portáveis para outros sistemas de programação/processamento paralelo ou distribuído, como o *nCUBE*, *SP1* ou *MPI*;
- Abrangência no emprego dos paradigmas de programação concorrente, como as construções dos modelos *SPMD*, *Master/Slave*, Síncrono e Assíncrono, e *Hosted*; proposta de um novo modelo de programação, o *Pipeline*, bastante adequado a sistemas de tempo-real, como as funções de análise de segurança de um Centro de Supervisão e Controle;
- Investigação da viabilidade da programação de tolerância a falha de *host* num ambiente de computação distribuída, para funcionamento em tempo de execução;
- Realização de vários testes e simulações com implementações seqüenciais e distribuídas para validar os algoritmos elaborados.

Conclusões sobre os Resultados Obtidos com as Implementações Seqüenciais

- Um par de contingências simples (por exemplo, A1 e A2), ambas críticas e localizadas distantes entre si, quando simuladas como uma contingência composta (i.e., dupla) resulta:

  - o método pela máxima violação (Stott), além de gerar maior quantidade de restrições, realiza mais iterações e tende a consumir maior tempo de computação se comparado ao método de Benders para a mesma condição de teste (Tabelas 6.7 e 6.8);
  - a razão para estes comportamentos distintos é atribuída aos cortes por multiplicadores simplex, geralmente mais ricos em informação sobre o distúrbio que os cortes pela máxima violação;

- Dado um sistema-teste, para determinados níveis de carregamento (geração/demanda), sem geradores fictícios o processo de otimização simplesmente poderia ficar sem solução (Tabelas 6.4 e 6.5);

- Quanto mais carregado estiver o sistema, maior será o trabalho computacional necessário para resolver o problema e mais restrições serão adicionadas à base do PL;

- O valor ótimo da função objetivo e o número de restrições na base do PL decrescem à medida que aumenta a capacidade de remanejamento permitido; o mesmo não se pode afirmar sobre a quantidade total de $[MW]$ realocados, que pode aumentar ou decrescer com o crescimento de Δ (Tabelas 6.9 e 6.10, e Figura 6.4);

- Para o programa FPORS-II, dado Δ, o número de iterações de operação é sempre igual ao número de restrições geradas; neste caso, a geração da restrição está vinculada à análise da contingência e cada nova restrição é imediatamente aplicada ao processo PL de verificação do remanejamento (Figura 6.2, e Tabelas 6.9 e 6.10);

- Para valores elevados de Δ, para o FPORS-II e o FPORS-III, os correspondentes números de iterações de operação (e do caso-base também), tendem a ser coincidentes em ambos os programas; a explicação é que o número de iterações de operação depende exclusivamente de quantas contingências críticas são tratadas;

- Para $\Delta = 0$ e dado um sistema elétrico, dentre os três programas elaborados o que apresenta os menores tempos de computação é o FPORS-I (é uma implementação especializada no cálculo sem capacidade corretiva pós-contingência; Tabela 6.11);

- Tomando-se a estação *SPARC2* como referência, as velocidades relativas de processamento das máquinas *SPARC4* e *SPARC ULTRA* são aproximadamente 2,2 e 5,4; estes números acompanham de perto as relações das correspondentes freqüências de *clock*;

- Através de testes com os sistemas *IEEE*-118 e BR-725, as comparações de desempenho dos programas FPORS-II e FPORS-III conduzem às seguintes conclusões:
  - para valores pequenos de capacidade de remanejamento (por exemplo, 0,001% a 3%), o FPORS-III que gera cortes densos apresenta os menores tempos de computação;
  - para valores grandes de capacidade de remanejamento (por exemplo, 8% a 40%), o desempenho é mais favorável ao FPORS-II (que exibe os menores tempos de computação);

  - à medida que o remanejamento cresce, para um mesmo sistema-teste, o tempo de computação do programa FPORS-II tende a aproximar-se do tempo obtido com o FPORS-I (compare as Tabelas 6.11 e 6.21);
  - em geral, para Δ pequeno (ou próximo de zero) é mais provável a necessidade de gerar mais cortes, embora a solução possa evoluir mais rapidamente para a solução final;
  - para listas com contingências múltiplas entre os casos críticos, o método de Benders mostra melhores desempenhos;

- Dado que o programa FPORS-I é sempre mais rápido que as outras duas implementações, porém só trabalha com $\Delta = \mathbf{0}$, surge uma interessante idéia de se elaborar um programa híbrido. Tal implementação associaria as diferentes metodologias de geração de cortes e, de forma seletiva em função da margem de remanejamento, seria capaz de tratar toda a gama de capacidade corretiva pós-contingência, tendo como inicializador a implementação mais simples (FPORS-I); as contingências críticas seriam aquelas que geraram restrições para a base do processo PL inicializador.

Conclusões sobre os Resultados Obtidos com as Implementações Distribuídas

- O tempo de inicialização do *PVM3* cresce com o aumento do número de nós alocados na máquina paralela virtual; o tempo de *I/O* depende do banco de dados do sistema sob análise e pode ser comparável aos tempos de execução de partes paralelizáveis de uma aplicação (vide Tabela 6.23); estes tempos não são importantes no cálculo do desempenho paralelo de aplicações práticas *on-line*;

- Para sistemas como o Sul-Sudeste do Brasil (por exemplo, o BR-725), os tempos consumidos no cálculo do FPO caso-base são pequenos se comparados às partes intensivas dos programas de FPORS;

- Na rede *FDDI* observou-se uma saturação dos ganhos paralelos (i.e., *speedups*) para um número relativamente pequeno de estações; os melhores ganhos relativos foram observados na rede *Ethernet* (compare a 6ª coluna das Tabelas 6.23, 6.24 e das Tabelas 6.25 e 6.26); a razão desta saturação se deve principalmente ao fato de que os sistemas baseados em fibras ópticas apresentam seus melhores desempenhos quando são intensivamente solicitados através de comunicações com elevadas quantidades de dados;

- Os *speedups* obtidos com sistemas-testes com listas longas (i.e., granularidade alta) são muito bons tendo em vista as características da aplicação e do ambiente de programação distribuída (por exemplo, conforme ilustra a Figura 6.6, o ganho alcançado com 5 estações é 3,5; para 2 e 3 estações os ganhos aproximam-se do ideal); a máxima eficácia é atingida com 4 estações;

- Os sistemas testados cujas listas de contingências são curtas (100 casos) ou médias (200 casos) apresentam *speedups* aceitáveis para 2 e 3 estações na máquina paralela virtual (vide Figura 6.6 e Tabela 6.29);

- Contar com a lista de contingências previamente ordenada na base de dados se traduz numa substancial economia de tempo de computação no cálculo do FPORS; a situação mais desfavorável em consumo de tempo e em ganhos paralelos é quando a lista não está ordenada e não se faz uma classificação prévia para identificar os casos críticos de sobrecarga;
- O nível da capacidade de remanejamento pós-contingência influencia o desempenho paralelo dos programas FPORS-II e III; a causa principal reside no desbalanço de carga entre processadores que é função do valor de Δ;
- O uso compartilhado de estações pode comprometer seriamente o desempenho de uma implementação distribuída; neste contexto, é importante o uso de propriedades dinâmicas (no balanceamento automático de cargas computacionais) que alguns sistemas apresentam, como é o caso do *PVM3*;
- A técnica de marcas, usada inicialmente nas implementações seqüenciais, foi estendida com sucesso aos programas paralelos com o controle de convergência global centralizado no processo do *host-login*.

Conclusões sobre o Modelo *Pipeline*

- O modelo *Pipeline* realmente é adequado ao processamento distribuído de sistemas de tempo-real, que são alimentados ininterruptamente por novas bases de dados;
- O *Pipeline* relativamente ao *Hosted* atrasa a saturação dos tempos de computação à medida que mais máquinas participam do processamento;
- Para se obter bons desempenhos do *Pipeline* deve ser observado o compromisso entre carga computacional das tarefas e número de processadores alocados para executá-las;
- Uma interessante característica do *Pipeline* é que os estágios que executam diferentes funções (caso-base, seleção e avaliação de contingências) operam de forma quase desacoplada graças ao assincronismo; isto repercute em *throughputs* superiores aos tradicionais modelos (como o *Hosted*);
- Cada estágio do *Pipeline* pode ser elaborado de maneira praticamente independente e comporta paradigmas complexos de programação concorrente;
- O *PVM3* permite a implementação de recursos de tolerancia a falha, de maneira relativamente simples.

Avaliação Crítica do Modelo *Pipeline*

Apesar do superior desempenho do *Pipeline* em relação ao *Hosted*, a saturação nas curvas tempo versus número de estações, para além de 4 máquinas, ainda pode ser observada em

algumas situações testadas. Ora, a saturação é um fenômeno freqüentemente observado em resultados práticos do emprego do processamento paralelo ou distribuído. No entanto, a maioria das pesquisas, a exemplo desta, busca responder a uma pergunta formulada há algum tempo: qual é o número ótimo (ou ideal) de processadores para um dado problema (de tamanho definido) executando numa certa arquitetura? Um dos pontos chaves deste trabalho consiste em apresentar uma alternativa vantajosa de paradigma de programação concorrente quando se deseja analisar ininterruptamente um fluxo de dados. Neste aspecto em particular, apresenta-se o *Pipeline* como opção vantajosa sobre o tradicional modelo *Hosted*.

Outra questão interessante de ser levantada é a seguinte: onde está o gargalo? No caso específico do *PVM3* configurando uma rede *LAN* de estações, apesar do processamento ser em paralelo, a comunicação se faz de forma seqüencial. Buscando-se responder a esta questão, foram avaliadas as prováveis influências da comunicação sobre o desempenho através da implementação de três formas de distribuir tarefas a partir de um *pool* de contingências: (1) o modo síncrono; (2) assíncrono por unidade, onde os casos são retirados um a um do *pool* de contingências; e (3) o modo assíncrono por "envelope". Em qualquer uma das modalidades descritas, o processo *master* também calcula contingências enquanto coordena a distribuição de tarefas. A idéia neste caso consiste em testar diferentes graus de utilização da rede de comunicação. Assim, (1) é a modalidade que menos usa a rede, mas pressupõe tarefas balanceadas; (2) utiliza com mais freqüencia a rede que (3); e as modalidades (2) e (3) pressupõem desbalanço de tarefas. Os testes demonstraram que os melhores desempenhos para este tipo de problema são alcançados com o modo síncrono (vide Figuras 6.14, 6.15, 6.26 e 6.27).

Afinal, qual é o número ideal de processadores? A própria filosofia do modelo *Pipeline* sugere que deve existir uma quantidade limite e ideal de processadores para executar um volume de trabalho especificado. Este número atende ao compromisso entre a redução de tempo em relação ao caso seqüencial e a contenção causada em vista do elevado número de processos operando sobre as tarefas. Ao ultrapassar este limiar, agregar mais "operários" sem aumentar o volume de trabalho seguramente chega-se à saturação. Contudo, a despeito de qualquer aspecto negativo que possa ser observado, os tempos de computação do *Pipeline* são significativamente menores (em alguns casos, da ordem de 60 %) que aqueles medidos no modelo *Hosted*.

O *Pipeline* funciona exatamente como uma linha de montagem em uma fábrica. Então, o que se observa para uma linha de montagem observa-se também no paradigma *Pipeline*. Um exemplo disso é o seguinte fato (vide Figuras 6.20 e 6.21): se com 4 estações consegue-se um bom desempenho (um *host* para as funções F_1 e F_4, dois para F_2, e outro para F_3; vide Figura 5.2 do Capítulo 5) ao se incrementar a configuração de uma máquina desequilibra-se o *Pipeline*, implicando num desempenho para 5 similar (ou até pior) ao conseguido com 4. Porém, agregando-se duas máquinas (e não uma como no exemplo anterior) fica-se com 6, que repercute numa ótima *performance*, uma vez que o *throughput* atinge a condição ideal pelo restabelecimento do equilíbrio. Conclui-se a partir dessa exposição que, é mais acentuado o efeito negativo do descasamento em *throughput* dos diversos estágios (portanto, o desequilíbrio) sobre o desempenho que a influência da etapa seqüencial do processamento.

Conclusões Gerais e Propostas de Trabalhos Futuros

(1) Este trabalho comprova que é viável usar computadores de alto desempenho configurados como ambientes de processamento distribuído na análise de sistemas elétricos de grande porte em aplicações de tempo-real; entretanto, acredita-se que, para esta classe de aplicações, o emprego do processamento distribuído deverá situar-se em configurações de 2 a 5 computadores;

(2) Ao paralelizar um dado algoritmo, uma correta decomposição do problema em tarefas e subtarefas é essencial a fim de se alcançar um bom desempenho;

(3) A programação estruturada e modularizada facilita a paralelização de aplicações;

(4) Uma característica da arquitetura distribuída (rede de estações com *PVM*) é que ela não requer estações dedicadas para o processamento de uma aplicação. Assim sendo, várias aplicações podem executar de forma concorrente na máquina paralela virtual, ou seja, as máquinas funcionam em um esquema de partilha de tempo. Entretanto, acredita-se que o uso compartilhado não é de interesse para aplicações práticas *on-line* do processamento distribuído em Centros de Controle;

(5) Sistemas de processamento distribuído como o *PVM3* mostram-se bastante promissores para o tratamento de problemas combinatoriais segundo a metodologia *A-Teams* (times assíncronos);

(6) Na atualidade, tanto os sistemas concorrentes quanto suas aplicações apresentam uma clara tendência para os ambientes de processamento distribuído; isto deverá conduzir a uma padronização de linguagens e métodos;

(7) A meta final desta linha de pesquisa é um programa que incorpore todos os controles relacionados ao fluxo de potência ótimo-seguro e que seja portável entre arquiteturas distintas; para atingir esta meta, a formulação do problema de otimização também deverá incluir a representação de variáveis e restrições do subproblema reativo; com a implementação da abordagem por decomposição de Benders este objetivo fica um pouco mais próximo;

(8) Outros métodos de análise de contingências, como, por exemplo, os que utilizam conceitos de vizinhanças, merecem ser pesquisados;

(9) Uma vantagem observada durante o desenvolvimento das implementações computacionais é que as técnicas iterativas de *screening* são mais fáceis de programar que as técnicas associadas aos métodos diretos. Embora, a complexidade cresça à medida que se incluem ordenações ou outros recursos para aprimorar o pré-condicionador;

(10) Tendo em vista o excelente desempenho dos métodos iterativos, e a relativa facilidade de programá-los para análise de contingências, abre-se um extraordinário campo para investigação; por exemplo, incorporação dos controles nos estados pós-contingência e pré-condicionadores baseados na formulação do fluxo de potência pelo método de Newton.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] TINNEY, W. F. and WALKER, J. W. Direct solution for sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, v.55, n.11, p.1801-1809, Nov. 1967.

[2] DOMMEL, H. W. and TINNEY, W.F. Optimal power flow solutions. *IEEE Transactions on Power Apparatus and Systems*, v.87, p.1866-1876, 1968.

[3] LASDON, L.S. **Optimization Theory for Large Systems.** NY: Macmillan, 1970. 523p.

[4] ZOLLENKOPF, K. Bi-fatorization basic computation algorithm and programming techniques. In: *REID, J. K. ed. Large sparse sets of linear equations.* New York: Academic Press, 1971, p.75-97.

[5] TINNEY, W.F. Compensation methods for network solutions by optimally ordered triangular factorizations. *IEEE Trans. on PAS*, v.PAS-91, p.123-127, Jan./Feb. 1972.

[6] GEOFFRION, A.M. Generalized Benders decomposition. *JOTA*, v.10, n.4, p.237-261, 1972.

[7] ALSAÇ, O. and STOTT, B. Optimal load flow with steady-state security. In: IEEE POWER INDUSTRY COMPUTER APPLICATIONS CONFERENCE, 1974. **PICA Proceedings...** p.745-751.

[8] STOTT, B. and ALSAÇ, O. Fast decoupled load flow. *IEEE Trans. on PAS*, v.93, p.859-869, May/June 1974.

[9] STOTT, B. and HOBSON, E. Power system security control calculations using linear programming, Part I. *IEEE Trans. on PAS*, v.97, n.5, p.1713-1720, Sept./Oct. 1978.

[10] STOTT, B. and HOBSON, E. Power system security control calculations using linear programming, Part II. *IEEE Trans. on PAS*, v.97, n.5, p.1721-1731, Sept./Oct. 1978.

[11] EJEBE, G.C. and WOLLENBERG, B.F. Automatic contingency selection. *IEEE Trans. on PAS*, v.PAS-98, p.92-104, Jan./Feb. 1979.

[12] STOTT, B. and MARINHO, J. L. Linear programming for power system security applications. *IEEE Trans. on PAS*, v.PAS-98, n.1, p.837-848, Jan. 1979.

[13] STOTT, B., MARINHO, J. L. and ALSAÇ, O. Review of linear programming applied to power system rescheduling. In: IEEE POWER INDUSTRY COMPUTER APPLICATIONS CONFERENCE, 1979, Cleveland. **PICA Proceedings...** May 1979, p.142-154.

[14] IRISARRI, G., SASSON, A. M. and LEVNER, D. Automatic contingency selection for on-line security analysis - real-time tests. *IEEE Trans. on PAS*, v.PAS-98, n.5, p.1552-1559, Sept./Oct. 1979.

[15] DY LIACCO, T. E. Mathematical challenges in electric power system operation: an overview. In: Electric Power Problems - The Mathematical Challenge, SIAM, 1980.

[16] MIKOLINNAS, T.A. and WOLLENBERG, W.F. An advanced contingency selection algorithm. *IEEE Trans. on PAS*, v.PAS-100, n.2, p.608-617, Feb. 1981.

[17] ALBUYEH, F., BOSE, A. and HEALTH, B. Reactive power considerations in automatic contingency selection. *IEEE Trans. on PAS*, v.101, n.1, p.608-617, Jan. 1982.

[18] WRUBEL, J. N. *et al.* Expanding an energy control center to include a bulk system security package. *IEEE Trans. on PAS*, v.PAS-101, n.1, p.34-42, Jan. 1982.

[19] ENNS, M.K., QUADA, J.J. and SACCKETT, B. Fast linear contingency analysis. *IEEE Trans. on PAS*, v.PAS-101, n.4, p. 783-791, April 1982.

[20] MONTICELLI, A. **Fluxo de Carga em Redes de Energia Elétrica.** São Paulo: Edgard Blücher, 1983. 164p.

[21] ALSAÇ, O., STOTT, B. and TINNEY, W.F. Sparsity-oriented compensation methods for modified network solutions. *IEEE Trans. on PAS*, v.102, p.1050-1060, May 1983.

[22] TINNEY, W.F., BRANDWAJN, V. and CHAN, S. M. Sparse vector methods. *IEEE Trans. on PAS*, v. PAS-104, p.295-301, Feb. 1985.

[23] PEREIRA, M.V.F., MONTICELLI, A. and PINTO, L.M.V.G. Security-constrained dispatch with corrective rescheduling. In: IFAC SYMPOSIUM ON PLANNING AND OPERATION OF ELECTRIC ENERGY SYSTEMS, 1985, Rio de Janeiro. **Proceedings...** 1985, p. 387-394.

[24] CHAN, S.M. and BRANDWAJN, V. Partial matrix refactorization. *IEEE Trans. on Power Systems*, v.PWRS-1, n.1, p.193-200, Feb. 1986.

[25] WU, F. F. and MONTICELLI, A. Recent progress in real-time network security analysis. In: *IFAC SYMPOSIUM ON POWER SYSTEMS AND POWER PLANT CONTROL*, Aug. 1986, Beijing, China. **Proceedings...** 1986, p.10-16.

[26] MONTICELLI, A., PEREIRA, M.V.F. and GRANVILLE, S. Security-constrained optimal power flow with post-contingency rescheduling. *IEEE Trans. on Power Systems*, v.PWRS-2, pp. 175-182, Feb. 1987.

[27] STOTT, B., ALSAÇ, O. and MONTICELLI, A. Security analysis and optimization. Invited Paper. *Proc. IEEE*, v.75, n.12, p.1623-1644, Dec. 1987.

[28] BETANCOURT, R. An efficient heuristic ordering algorithm for partial matrix refactorization. *IEEE Trans. on Power Systems*, v.3, n.3, p.1181-1187, Aug. 1988.

[29] GÓMEZ, A. and FRANQUELO, L.G. An efficient ordering algorithm to improve sparse vector methods. *IEEE Trans. on Power Systems*, v.3, n.4, p.1538-1544, Nov. 1988.

[30] LO, K.L., ARSHAD, B., McCOLL, R.D. and MOFFAT, A.M. A comparison of MW ranking methods. *Electric Power Systems Research*, 15, p.157-171, 1988.

[31] DUNCAN, R. A survey of parallel computer architectures. *Computer*, v.23, n.2, p.5-16, Feb. 1990.

[32] MOONEY, J.D. Strategies for supporting application portability. *Computer*, v.23, n.11, p.59-70, Nov. 1990.

[33] PINTO, H.J.C.P., PEREIRA, M.V. and TEIXEIRA, M.J. New parallel algorithms for the security-constrained dispatch with post-contingency corrective actions. In: POWER SYSTEM COMPUTATION CONFERENCE, X, 1990, Graz. **PSCC Conf. Proc.**, 1990.

[34] TEIXEIRA, M. J., PINTO, H. J. C. P., PEREIRA, M. V. and McCOY, M. F. Developing concurrent processing applications to power system planning and operations. *IEEE Trans. on Power Systems*, v.5, n.2, p.659-664, May 1990.

[35] BAZARAA, Mokhtar S. *et al.* **Linear Programming and Network Flows.** New York: John Wiley & Sons, 2nd ed., 1990. 684p.

[36] HUNEAULT, M. and GALIANA, F. D. A survey of the optimal power flow literature. *IEEE Trans. on Power Systems*, v.6, n.2, p.762-770, May 1991.

[37] BAILEY, D.H. Twelve ways to fool the masses when giving performance results on parallel computers. Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center, Moffett Field, California. Technical Report. 5p. June 1991.

[38] TALUKDAR, S. *et al.*. Multiagent organizations for real-time operations. *Proc. IEEE*, v.80, n.5, p.765-778, May 1992.

[39] nCUBE **nCUBE 2 Programmer's guide - release 3.0 version.** California, 1992.

[40] BALU, N. *et al.* On-line power system security analysis. Invited Paper. *Proceedings of the IEEE*, v.80, n.2, p.262-280, Feb. 1992.

[41] TYLAVSKY, D.J., BOSE, A. *et al.* Parallel processing in power systems computation. *IEEE Trans. on Power Systems*, v.7, n. 2, p.629-638, May 1992.

[42] PINTO, D.P. e PEREIRA, J.L.R. Um método localizado para análise de contingências estáticas em sistemas de energia elétrica. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, IX, 1992, Vitória, Brasil. **Anais...** pp.604-608.

[43] MÉNDEZ, O. R. Saavedra. **Solução concorrente do problema do fluxo de potência ótimo com restrições de segurança.** Campinas, Brasil: Faculdade de Engenharia Elétrica (UNICAMP), 1993. 106p. (Tese, Doutorado).

[44] SANTOS, José V.C. **Análise de segurança estática de sistemas de potência: um estudo sobre critérios de seleção de contingências no subproblema reativo.** Campinas: FEE, UNICAMP, 1993. Dissertação (Mestrado) - Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, 1993. p.36-50.

[45] GALIANA, F.D., JAVIDI, H. and McFEE, S. On the application pre-conditioned conjugate gradient algorithm to power network analysis. In: IEEE PICA CONFERENCE, May 1993. **PICA Proceedings...** 1993, p.404-410.

[46] CALZAROSSA, M. and SERRAZI, G. Workload characterization: a survey. *Proc. IEEE.* v.81, No. 8, p.1136-1150, Aug. 1993.

[47] ALVES, A.C.B. **Uma introdução à programação paralela.** UNICAMP. Relatório Técnico Interno. 48 p. agosto 1993.

[48] DOUGLAS, C.C. *et al.* **Parallel programming systems for workstation clusters.** Yale University Department of Computer Science Research. Technical Report. - 36p. Aug. 1993.

[49] CHOUDHARY, A.N. *et al.* Optimal processor assignment for a class of pipelined computations. *IEEE Trans. on Parallel and Distributed Systems*, v.5,n.4, p.439-445, April 1994.

[50] TALUKDAR, S.N. A multi-agent technique for contingency constrained optimal power flows. *IEEE Trans. on Power Systems*, v.9, n.2, p.855-861, May 1994.

[51] HATZIARGYRIOU, N.D., CONTAXIS, G.C. and SIDERIS, N.C. A decision tree method for on-line steady state security assessment. *IEEE Trans. on Power Systems*, v.9, n.2, p.1052-1061, May 1994.

[52] GEIST, A. *et al.* **PVM 3 User's guide and reference manual.** Oak Ridge: Oak Ridge National Laboratory, May 1994. (ORNL/TM-12187).

[53] ALVES, A.C.B. **PVM3: Um *software* para processamento paralelo distribuído.** UNICAMP. Relatório Técnico Interno. 92 p. julho 1994.

[54] MELIOPOULOS, A.P., CHENG, C.S. and XIA, F. Performance evaluation of static security analysis methods. *IEEE Trans. on Power Systems*, v.9, n.3, p.1441-1449, Aug. 1994.

[55] PINTO, D.P. e PEREIRA, J.L.R. Um método integrado para análise de contingências utilizando processamento paralelo. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, X, setembro 1994, Rio de Janeiro, Brasil. **Anais...** p.179-184.

[56] RODRIGUES, M., SAAVEDRA, O.R. and MONTICELLI, A. Asynchronous programming model for the concurrent solution of the security constrained optimal power flow problem. *IEEE Trans. on Power Systems*, v.9, n.4, p.2021-2027, Nov.1994.

[57] FALCÃO, D.M. Parallel and distributed processing applications in power system simulation and control. In: Revista da Sociedade Brasileira de Automática, v.5, Nº Único, Edição Especial, Mini-Cursos do 10º CBA e 6º Congresso Latino Americano de Controle Automático, Rio de Janeiro, p.125-143, Oct./Nov. 1994.

[58] VLACHOGIANNIS, J.G. Control adjustments in fast decoupled load flow. *Electric Power Systems Research*, 31, p.185-194, 1994.

[59] WOOD, A. J. and WOLLENBERG, B. F. **Power Generation Operation and Control.** New York: John Wiley & Sons, 2nd ed., 1996. 569p.

[60] ALVES, A.C.B. **Soluções paralela e distribuída do problema de análise de contingências *on-line* nos modos síncrono e assíncrono.** UNICAMP. Relatório Técnico Interno. 98p. janeiro 1995.

[61] HAO, S., PAPALEXOPOULOS, A. and PENG, TIE-MAO. Distributed processing for contingency screening applications. *IEEE Trans. on Power Systems*, v.10, n.2, p.838-844, May 1995.

[62] MORI, H., TANAKA, H. and KANNO, J. A preconditioned fast decoupled power flow method for contingency screening. In: IEEE POWER INDUSTRY COMPUTER APPLICATIONS CONFERENCE, May 1995. **PICA Proceedings...** 1995.

[63] CASTRO, M. Stehling. **Análise *on-line* da estabilidade transitória de sistemas elétricos de potência usando computação distribuída.** Campinas, Brasil: Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas (UNICAMP), 1995. 98p. (Dissertação, Mestrado).

[64] SATO, Fujio **Um estudo comparativo da análise de curto-circuito probabilístico em ambientes paralelo e distribuído.** Campinas, Brasil: Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas (UNICAMP), 1995. 170p. (Tese, Doutorado).

[65] ZHANG, Y. and TINNEY, W.F. Partial refactorization with unrestricted topology changes. *IEEE Trans. on Power Systems*, v.10, n.3, p.1361-1368, Aug. 1995.

[66] ALVES, A.C.B. **Métodos iterativos aplicados à solução do problema de *screening* de contingências em redes elétricas.** UNICAMP. Relatório Técnico Interno. 75p. janeiro 1996.

[67] RAMESH, V.C. and TALUKDAR, S.N. A parallel asynchronous for decomposition on-line contingency planning. *IEEE Trans. on Power Systems*, v.11, n.1, p.344-349, Feb. 1996.

[68] EKMECIĆ, I., TARTALJA, I., and MILUTINOVIĆ, V. A survey of heterogeneous computing: concepts and systems. *Proc. IEEE*, v.84, n.8, p.1127-1144, Aug. 1996.

Apêndice A

Algoritmo de Programação Linear Utilizado

O algoritmo especializado de solução de PL que está descrito neste apêndice corresponde à versão documentada no artigo de B. Stott e J.L. Marinho [12], resultante de melhoramentos de trabalhos precedentes [9,10]. Na Figura 2.3 estão ilustradas, por meio de blocos, as etapas principais deste algoritmo.

Preliminares

O algoritmo de programação linear utilizado neste trabalho resolve um problema de otimização típico de redes elétricas, que pode ser representado da seguinte maneira:

$$\begin{cases} \text{Min} \quad f(\mathbf{\Delta P}) = \mathbf{c\Delta P} \\ \text{s.a.} \quad \beta^t \mathbf{\Delta P} = 0 \\ \qquad \Delta P_j^{\text{mín}} \leq A_j^t \mathbf{\Delta P} \leq \Delta P_j^{\text{máx}} \quad \text{para todo gerador } j \\ \qquad \Delta f_{km}^{\text{mín}} \leq A_{km}^t \mathbf{\Delta P} \leq \Delta f_{km}^{\text{máx}} \quad \text{para todo ramo } k-m \end{cases} \tag{A.1}$$

Onde,

β é o vetor de ordem $(n \times 1)$, que armazena os fatores de perdas na transmissão (se forem desprezadas as perdas, todos os $\beta_j = 1$);

A_j é um vetor restrição esparso com 1 na posição j;

A_{km} é um vetor restrição não-esparso, que pode ser obtido resolvendo-se a equação seguinte

para cada ramo $k - m$:

$$[B']A_{km} = \begin{array}{|c|c|c|c|c|c|c|} \hline & & k & & m & & t \\ \hline 0 & \cdots & -b_{km} & \cdots & +b_{km} & \cdots & 0 \\ \hline \end{array}$$

Na expressão (A.1), $\mathbf{c}$ é o vetor de custos incrementais e $\mathbf{\Delta P} = (\Delta P_1 \cdots \Delta P_n)^t$ as variáveis de decisão do processo PL; os limites das restrições são postos na forma incremental, portanto, o primeiro passo é estabelecer o modelo incremental das variáveis e equações da rede. Isto pode ser feito através de um cálculo de fluxo de potência linearizado.

Do cálculo do fluxo de potência inicial obtém-se θ e do banco de dados tem-se a geração inicial, $\mathbf{P}^{\text{inicial}}$. O procedimento para estabelecer a forma incremental dos limites das variáveis é o seguinte:

$$\begin{aligned} f_{km}^{\text{inicial}} &= -b_{km}(\theta_k - \theta_m) \\ \Delta f_{km}^{\text{máx}} &= f_{km}^{\text{máx}} - f_{km}^{\text{inicial}} \\ \Delta f_{km}^{\text{mín}} &= f_{km}^{\text{mín}} - f_{km}^{\text{inicial}} \\ \Delta P_j^{\text{máx}} &= P_j^{\text{máx}} - P_j^{\text{inicial}} \\ \Delta P_j^{\text{mín}} &= P_j^{\text{mín}} - P_j^{\text{inicial}} \end{aligned} \tag{A.2}$$

Se todas as restrições da rede fossem calculadas *a priori* e se o PL fosse montado completo, o problema ficaria enorme[1] (número de restrições igual a $1 + n + n^{\underline{o}}$ de ramos e n variáveis). O algoritmo de solução de PL de Stott aborda o problema (A.1), primeiramente, relaxando todas as restrições de ramo. É o chamado despacho inicial relaxado, que corresponde ao PL (A.3).

$$\left\{ \begin{array}{lll} \text{Min} & f(\mathbf{\Delta P}) = \mathbf{c\Delta P} & \\ \text{s.a.} & \beta^t \mathbf{\Delta P} = 0 & \\ & \Delta P_1^{\text{mín}} \le \Delta P_1 \le \Delta P_1^{\text{máx}} & \\ & \Delta P_2^{\text{mín}} \le \Delta P_2 \le \Delta P_2^{\text{máx}} & \\ & \qquad\qquad \ddots & \\ & \Delta P_n^{\text{mín}} \le \Delta P_n \le \Delta P_n^{\text{máx}} & \end{array} \right. \tag{A.3}$$

[1] Este é um aspecto notável do algoritmo: a cada passo apenas poucos ramos apresentarão violações dos seus limites de fluxos.

O problema (A.3) é resolvido do seguinte modo: coloca-se $\Delta P^{\text{mín}}$ em todas as barras e aumenta-se a geração em ordem crescente de custo incremental, até que se obtenha o balanço de potência ($\sum_{j=1}^{n} \beta_j \Delta P_j = 0$). Tem-se, então, alguns geradores no limite superior, outros no limite inferior e um deles ajustado em um valor intermediário entre os limites superior e inferior (isto é, livre). A base inicial é formada por este conjunto de restrições ativas mais a equação de balanço.

Neste estágio do algoritmo, alguns ramos estarão com seus limites de fluxos violados; isto pode ser verificado através do estado incremental $\mathbf{\Delta\theta}$ obtido da solução de $[B']\mathbf{\Delta\theta} = \mathbf{\Delta P}$, assim:

$$\Delta f_{km} = -b_{km}(\Delta\theta_k - \Delta\theta_m) \tag{A.4}$$

$$\text{Critério heurístico} \begin{cases} \text{Se } \Delta f_{km} \geq 90\% \Delta f_{km}^{\text{máx}} \rightarrow k-m \text{ no conjunto monitorado} \\ \text{Se } \Delta f_{km} \leq 90\% \Delta f_{km}^{\text{mín}} \rightarrow k-m \text{ no conjunto monitorado} \end{cases}$$

Os ramos colocados no conjunto monitorado estão violados (limite superior ou inferior) ou estão na iminência de sofrerem violação.

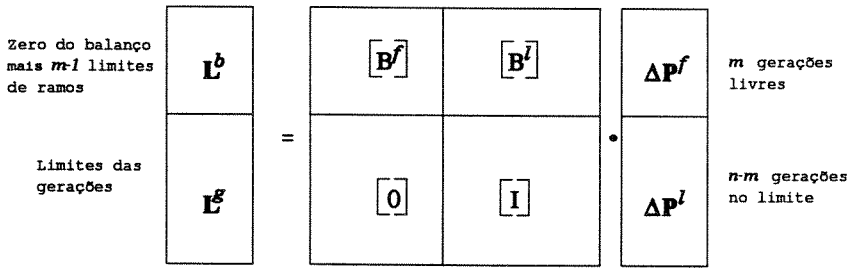
É importante ressaltar que o PL dado em (A.1) não apresenta explicitamente as variáveis de estado ($\mathbf{\Delta\theta}$). Sempre que se obtém um novo $\mathbf{\Delta P}$, seja no despacho inicial ou durante as iterações do algoritmo, calcula-se o estado pela expressão $[B']\mathbf{\Delta\theta} = \mathbf{\Delta P}$ para, em seguida, atualizar o conjunto monitorado (vide blocos F e B da Figura 2.3).

Após a obtenção de $\mathbf{\Delta\theta}$ e dos cálculos dos fluxos será constatado que alguns ramos estarão com seus limites violados. A cada iteração do algoritmo, o fluxo do ramo mais violado é fixado em seu limite (para ele, uma restrição do tipo $\mathbf{A}_{km}^{t}$ é construída), em troca, um gerador passa a ser livre. Se, em outra situação, algum gerador tiver seu limite violado, sua geração incremental será ajustada no limite e, conseqüentemente, um outro gerador ou ramo ficará livre. Esta é a idéia geral do algoritmo de solução de PL de Stott.

Metodologia de Otimização

Sendo $[B]$ a matriz base, o ponto de operação do sistema no início de cada iteração é dado pela equação básica:

$$\mathbf{L} = [B]\mathbf{\Delta P} \tag{A.5}$$



Onde, os superíndices b, g, f e l significam ***branch***, ***generation***, ***free*** e ***limited***, respectivamente; $[I]$ é uma matriz identidade; $\mathbf{L}$ é o vetor de limites das restrições ativas.

A primeira linha do sistema matricial (A.5) é a restrição de balanço de potência (permanentemente na base):

$$0 = (\beta_1\ \beta_2\ \cdots\ \beta_n)(\Delta\mathbf{P}^f \mid \Delta\mathbf{P}^l)^t \tag{A.6}$$

A obtenção dos fatores β_j será mostrada ao final deste apêndice.

As linhas das gerações ativas não são armazenadas; através de apontadores sabe-se, durante o processo, qual gerador é livre e qual está no limite. As matrizes $[B^f]$ e $[B^l]$ são armazenadas e, sempre que ocorre uma mudança na base, os apontadores e as matrizes são atualizados.

De (A.5) pode-se escrever a equação da base reduzida:

$$\mathbf{L}^b = [B^f]\Delta\mathbf{P}^f + [B^l]\Delta\mathbf{P}^l = [B^r]\Delta\mathbf{P} \tag{A.7}$$

As gerações livres podem ser expressas em função dos limites incrementais de fluxos e das gerações em limite:

$$\Delta\mathbf{P}^f = [B^f]^{-1}\{\mathbf{L}^b - [B^l]\Delta\mathbf{P}^l\} \tag{A.8}$$

A equação (A.8) é utilizada toda vez que ocorrer uma mudança de base. O esforço computacional para resolver esta equação é pequeno, entretanto, é importante fatorar $[B^f]$ (mesmo sendo de ordem baixa), inclusive com escolha do pivô adequado para prevenir problemas de mal-condicionamento numérico.

As gerações ou fluxos violados são fixados no limite correspondente ($\mathbf{\Delta P}^l$ e $\mathbf{L}^b$) pela introdução de restrições de igualdade na base de maneira seqüencial. Ao fixar uma geração ou fluxo, outra geração/fluxo que esteja no limite será liberada. A questão é, como escolher a variável para entrar na base e a variável para sair da base.

A restrição a ser removida da base (variável que sai da base) é selecionada segundo critérios de otimalidade. A seleção é realizada em dois passos:

(1) as restrições ativas existentes que estão na base são testadas quanto a elegibilidade:

> Uma restrição é elegível quando, liberada com o fim de aliviar a violação da restrição entrante, ela abandona o limite no qual estava fixada. Se não houver restrição elegível, o problema será infactível.

A restrição entrante mencionada acima é a restrição candidata a entrar na base, arbitrariamente escolhida a mais violada (vide blocos C e E da Figura 2.3).

(2) a escolha ótima dentre as restrições elegíveis é feita pelo teste da razão mínima; que corresponde a selecionar a menor razão $|\lambda_k/S_k|$, onde λ_k é o custo relativo (i.e., componente k do vetor dual λ) e S_k é a sensibilidade da restrição entrante em relação à restrição candidata k:

> A restrição ativa a ser liberada é aquela que, dentre todas as elegíveis, a razão $|\lambda_k/S_k|$ é mínima.

Do exposto, observa-se que é preciso calcular a cada iteração os vetores λ e $\mathbf{S}$. Supondo-se o vetor restrição entrante ($\mathbf{A}^t_{km}$ ou $\mathbf{A}^t_j$) particionado como $\mathbf{A}^t = (\mathbf{A}^t_f \mid \mathbf{A}^t_l)$, a sensibilidade também resulta particionada do seguinte modo:

$$\mathbf{S}^b = \mathbf{A}^t_f[B^f]^{-1} \tag{A.9}$$

$$\mathbf{S}^g = \mathbf{A}^t_l - \mathbf{S}^b[B^l] \tag{A.10}$$

Onde, $\mathbf{S}^b$ é o vetor de sensibilidades da restrição entrante para os ramos que estão no limite na base; $\mathbf{S}^g$, de modo análogo, refere-se às sensibilidades da restrição para os geradores fixados no limite.

O vetor λ de variáveis duais (i.e., custos marginais, multiplicadores simplex, etc.), particionado, é calculado conforme as expressões (A.11) e (A.12).

$$\lambda^b = \mathbf{c}_f[B^f]^{-1} \tag{A.11}$$

$$\lambda^g = \mathbf{c}_l - \lambda^b[B^l] \tag{A.12}$$

Onde, $\mathbf{c}_f$ e $\mathbf{c}_l$ são as partições do vetor de custos incrementais (vide (A.1)).

O cálculo de λ é também importante na construção de restrições para o método de Benders.

Conhecidos λ e $\mathbf{S}$, o teste de elegibilidade é efetuado facilmente pesquisando-se a menor razão $|\lambda_k/S_k|$ dentre os índices das restrições elegíveis:

$\lambda = (\lambda^b \mid \lambda^g) =$ | ... | λ_k | ... |

$\mathbf{S} = (\mathbf{S}^b \mid \mathbf{S}^g) =$ | ... | S_k | ... |

Mudança na base pode acontecer de quatro modos distintos, como a seguir:

1. uma geração livre se torna limitada (ativa) em troca de uma geração limitada que se torna livre; os apontadores são simplesmente trocados – a base reduzida não é afetada;
2. uma nova restrição de ramo ($\mathbf{A}^t_{km}$) é adicionada à base $[B^r]$; uma geração é liberada e seu apontador é simplesmente trocado;
3. uma nova restrição de ramo substitui uma anteriormente existente, isto é, $\mathbf{A}^t_{km}$ entra no lugar da restrição de ramo velha em $[B^r]$;
4. uma geração anteriormente livre entra na base em lugar de uma restrição de ramo ativa, cuja linha $\mathbf{A}^t_{km}$ é removida de $[B^r]$; o apontador da geração é mudado.

Em todas estas mudanças o vetor $\mathbf{L}$ é atualizado na posição correspondente.

Na Figura 2.3, o bloco D, designado por "Núcleo do processo PL dual-simplex revisado", compreende os passos relacionados a seguir, os quais se constituem no "coração" do algoritmo de solução de PL de Stott:

(i) calcular as sensibilidades da restrição entrante para os ramos/geradores que estão no limite, através de (A.9) e (A.10);

(ii) identificar o conjunto das restrições ativas que são elegíveis – elegibilidade;

(iii) calcular o vetor de custos marginais λ, aplicando-se as expressões (A.11) e (A.12);

(iv) aplicar o teste da razão mínima para selecionar a restrição a ser liberada;

(v) atualizar a equação básica (A.5);

(vi) atualizar as gerações livres ($\Delta \mathbf{P}^j$), aplicando-se (A.8).

Multi-segmentação da Função Objetivo

Um aspecto notável do algoritmo de solução de PL ora apresentado é a capacidade de permitir a consideração de funções objetivos quadráticas. Isto pode ser incorporado graças à representação da função quadrática, contínua e convexa, como uma função linear por partes. É a chamada multi-segmentação. Por exemplo, a curva ilustrada na Figura A.1(a) pode ser aproximada por 3 segmentos cujas declividades são s_{j1}, s_{j2} e s_{j3} [59]. No exemplo da Figura

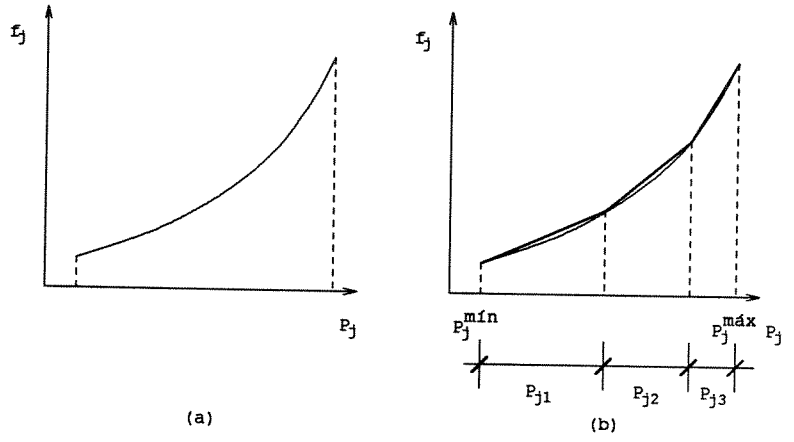


Figura A.1: A função custo do gerador j: (a) quadrática; (b) linearizada.

A.1 a função custo será representada pelas expressões

$$
\begin{aligned}
& f_j(P_j) = f_j(P_j^{\text{mín}}) + s_{j1}P_{j1} + s_{j2}P_{j2} + s_{j3}P_{j3} \\
& P_{j1}^- \leq P_{j1} \leq P_{j1}^+ \\
& P_{j2}^- \leq P_{j2} \leq P_{j2}^+ \\
& P_{j3}^- \leq P_{j3} \leq P_{j3}^+ \\
& P_j = P_j^{\text{mín}} + P_{j1} + P_{j2} + P_{j3}
\end{aligned}
$$

A função custo não-linear é, então, representada por uma expressão linear nos valores P_{j1}, P_{j2} e P_{j3}.

Regra geral, as declividades são calculadas através da expressão:

$$s_{jk} \triangleq \frac{f_j(P_{jk}^+) - f_j(P_{jk}^-)}{P_{jk}^+ - P_{jk}^-}$$

Onde, P_{jk}^+ e P_{jk}^- são os valores de P_j nos extremos dos segmentos lineares da j-ésima curva e k indica o número do segmento. A Figura A.2 ilustra o princípio da multi-segmentação empregado no processo PL de Stott.

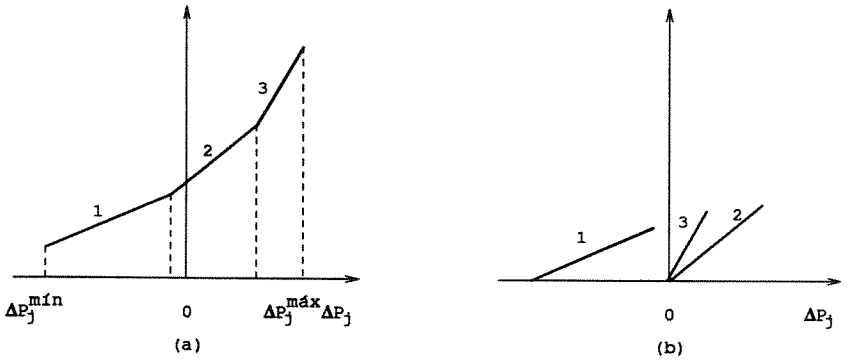


Figura A.2: Representação da curva de custo de um gerador como uma função linear por partes.

Cada segmento é representado como se fosse uma unidade de geração separada, conforme indica a Figura A.2(b), onde a componente de custo fixo é suprimida já que não é relevante no mecanismo de otimização. A multi-segmentação pode ser onerosa em termos de tempo de processamento e armazenamento se elevada exatidão na aproximação da curva for requerida. Além disso, a multi-segmentação exige um trabalho computacional extra quanto à pesquisa da elegibilidade [9,10,12,13].

Obtenção dos Fatores de Perdas na Transmissão

Na equação de balanço de potência ativa, as perdas na transmissão podem ser agrupadas numa única parcela, ΔP_L:

$$\Delta P_1 + \Delta P_2 + \cdots + \Delta P_n + \Delta P_L = 0 \qquad \text{(A.13)}$$

Do cálculo diferencial é possível expressar ΔP_L através da seguinte soma:

$$\Delta P_L = \frac{\partial P_L}{\partial P_1}\Delta P_1 + \frac{\partial P_L}{\partial P_2}\Delta P_2 + \cdots + \frac{\partial P_L}{\partial P_n}\Delta P_n \qquad \text{(A.14)}$$

Onde, P_L representa as perdas totais na transmissão.

Substituindo-se (A.14) em (A.13) e rearranjando a expressão resultante obtém-se a equação de balanço com as perdas, conforme (A.15).

$$(1 - \frac{\partial P_L}{\partial P_1})\Delta P_1 + (1 - \frac{\partial P_L}{\partial P_2})\Delta P_2 + \cdots + (1 - \frac{\partial P_L}{\partial P_n})\Delta P_n = 0 \qquad \text{(A.15)}$$

O coeficiente $(1 - \frac{\partial P_L}{\partial P_j})$ é designado por β_j e é conhecido como fator de perdas na transmissão. A questão é como calcular β_j? Sendo G_{km} os elementos da parte real da matriz $[Y]$, as perdas na transmissão para uma rede são obtidas através do somatório [20]:

$$P_L = \sum_{m \in \Omega_k} G_{km}\{(V_k^2 + V_m^2) - 2V_kV_m cos\theta_{km}\}$$

Onde, V_k e θ_k são a magnitude e o ângulo da tensão na barra k; Ω_k é o conjunto das barras vizinhas da barra k. Por outro lado, na forma matricial, a derivada das perdas em relação às injeções de potência ativa nas barras é dada por:

$$\frac{\partial P_L}{\partial \mathbf{P}} = \left[\frac{\partial \boldsymbol{\theta}}{\partial \mathbf{P}}\right]^t \frac{\partial P_L}{\partial \boldsymbol{\theta}} = [H]^{-1}\frac{\partial P_L}{\partial \boldsymbol{\theta}}$$

Onde, $[H]$ é a submatriz jacobiana do fluxo de potência pelo método de Newton. Considerando-se a modelagem linear, $[H]$ pode ser substituída pela matriz $[B']$ do método desacoplado rápido.

Agora, é fácil verificar que um procedimento para calcular os elementos β_j, $j = 1, 2, \cdots, n$, do vetor β compõe-se dos seguintes passos:

1. Conhecidos o estado da rede e a matriz $[Y]$, avaliar as n componentes do vetor $\frac{\partial P_L}{\partial \theta}$ por meio da expressão:

$$\text{Componente } k \rightarrow \frac{\partial P_L}{\partial \theta_k} = 2 \sum_{m \in \Omega_k} V_kV_mG_{km}sen\theta_{km}$$

2. Resolver o sistema linear $\frac{\partial P_L}{\partial \mathbf{P}} = [B']^{-1}\frac{\partial P_L}{\partial \theta}$;

3. Da solução obtida no passo 2, cada componente j de $\frac{\partial P_L}{\partial \mathbf{P}}$ fornece o β_j procurado, assim:

$$\beta_j = 1 - \frac{\partial P_L}{\partial P_j}$$

De posse dos valores dos fatores de perdas na transmissão tem-se a primeira restrição do PL:

$$\beta_1\Delta P_1 + \beta_2\Delta P_2 + \cdots + \beta_n\Delta P_n = \beta^t\Delta P = 0$$

Apêndice B

Decomposição de Benders

Decomposição ou partição de Benders é um método que se aplica à solução de problemas de otimização mistos. Problemas mistos são tipos especiais de problemas, geralmente de grande porte, que possuem variáveis simples (por exemplo, contínuas) e também variáveis complicantes (por exemplo, pertencentes a um subconjunto dos números reais ou de difícil determinação).

A técnica de decomposição de Benders aplica-se tanto a problemas lineares mistos quanto a problemas não-lineares mistos. A referência [6] generaliza a aplicação desta técnica; outras referências também trazem esclarecimentos importantes neste tema [3,23].

Nesta abordagem é mostrada a partição de Benders aplicada a problemas lineares onde há dois tipos de variáveis, representadas pelos vetores $\mathbf{x_0}$ e $\mathbf{x}$.

O Problema e o Conceito de Projeção

Seja o problema linear (B.1) nas variáveis $\mathbf{x_0}$ e $\mathbf{x}$:

$$\begin{cases} z = \text{Min} & \mathbf{c}\mathbf{x_0} \\ \mathbf{x_0} \in \mathcal{X}_0 & \\ \text{s.a.} & [A]\mathbf{x_0} \geq \mathbf{b} \\ & [E]\mathbf{x_0} + [F]\mathbf{x} \geq \mathbf{h} \end{cases} \tag{B.1}$$

Onde,

$\mathbf{x}$ é o vetor de variáveis reais;

$\mathbf{x_0}$ é o vetor de variáveis complicantes e $\mathcal{X}_0$ é o conjunto das variáveis x_{0i};

$\mathbf{b}$ e $\mathbf{h}$ são vetores de recursos, e $\mathbf{c}$ é o vetor de custos;

$[A]$, $[E]$ e $[F]$ são matrizes de coeficientes, de dimensões convenientes.

A forma dada em (B.1) está adaptada à formulação do FPORS [23]. Nesta equação, $[E]\mathbf{x}_0 + [F]\mathbf{x} \geq \mathbf{h}$ representa os subproblemas dos estados pós-contingências, incluindo as restrições de acoplamento $|\mathbf{x} - \mathbf{x}_0| \leq \boldsymbol{\Delta}$. Nota-se também que a função objetivo não possui variáveis $\mathbf{x}$.

Assume-se por simplicidade que (B.1) é factível e tem ótimo finito.

A abordagem proposta por Benders e Geoffrion para resolver (B.1) consiste em decompor o problema em dois subproblemas: um nas variáveis x_{0i} e o outro nas variáveis x_j. Projetar (B.1) sobre o espaço das variáveis x_{0i} significa fixar $\mathbf{x}_0$. Fixando-se o ponto $\mathbf{x}_0$, o subproblema de operação passa a ter o seguinte espaço de solução:

$$[F]\mathbf{x} \geq \mathbf{h} - [E]\mathbf{x}_0 \tag{B.2}$$

Factibilidade do primal e a relação com o dual:

Da teoria da dualidade de programação linear, sendo $\mathcal{T}$ o conjunto de vértices do poliedro convexo do problema dual associado a um dado problema primal, afirma-se o seguinte [35]:

- se o dual possuir solução ilimitada para algum $\mathbf{x}_0$ fixo, então o problema primal será infactível.

O problema mostrado em (B.2) é o subproblema primal. Para algum $\mathbf{x}_0 \in \mathcal{X}_0$ é possível que (B.2) seja infactível. Em lugar de se determinar o conjunto $\mathcal{V} = \{\mathbf{x}_0 \mid [E]\mathbf{x}_0 + [F]\mathbf{x} \geq \mathbf{h}$ para algum $\mathbf{x}\}$, é mais interessante procurar uma condição tal que (B.2) seja sempre factível.

Garantir que o subproblema primal é sempre factível, equivale a garantir que a solução ótima do dual é limitada.

Isto posto, o objetivo agora é garantir a "factibilidade" do subproblema primal (B.2) com $\mathbf{x}_0$ fixo. Introduz-se no espaço de solução variáveis artificiais (ou *slack*) $\mathbf{s} = (\, s_1\ s_2\ \cdots\ s_j\ \cdots)^t$:

$$\begin{array}{lll} \text{Antes} & [F]\mathbf{x} \geq \mathbf{h} - [E]\mathbf{x}_0, & \mathbf{x}_0 \text{ fixo} \\ \text{Depois} & [F]\mathbf{x} + [I]\mathbf{s} \geq \mathbf{h} - [E]\mathbf{x}_0, & \mathbf{x}_0 \text{ fixo}, \ \mathbf{s} \geq \mathbf{0} \end{array}$$

Assegurar que (B.2) seja sempre factível equivale a resolver o problema (B.3):

$$\left\{ \begin{array}{lll} w(\mathbf{x}_0) = \text{Min} & \sum_j s_j & \\ \text{s.a.} & [F]\mathbf{x} + [I]\mathbf{s} \geq \mathbf{h} - [E]\mathbf{x}_0 & \\ & \mathbf{s} \geq \mathbf{0} & \\ & \mathbf{x}_0 \ \text{ fixo} & \end{array} \right. \tag{B.3}$$

Então, pode-se estabelecer o seguinte critério [23,26]:

$$w(\mathbf{x}_0) = 0 \iff \mathbf{x}_0 \text{ é um ponto adequado}$$

$$w(\mathbf{x}_0) > 0 \iff \mathbf{x}_0 \text{ não é um ponto adequado}$$

O problema dual de (B.3) é Max $\{\mathbf{u}(\mathbf{h} - [E]\mathbf{x}_0)$ s.a. $\mathbf{u}[F] + \mathbf{u}[I] \leq \mathbf{d}, \mathbf{u} \in \mathcal{T}\}$, onde $\mathbf{d}$ é um vetor constituído por elementos 0 e 1. Os vértices do espaço factível deste problema são representados pelo conjunto $\mathcal{T} = \{\mathbf{u}_\tau^1, \mathbf{u}_\tau^2, \cdots, \mathbf{u}_\tau^p\}$.

É possível expressar $w(\mathbf{x}_0)$ em função dos vértices do problema dual de (B.3):

$$\left\{\begin{array}{lll} w(\mathbf{x}_0) = \text{Min} & \alpha & \\ \text{s.a.} & \mathbf{u}_\tau^1(\mathbf{h} - [E]\mathbf{x}_0) \leq \alpha & \\ & \mathbf{u}_\tau^2(\mathbf{h} - [E]\mathbf{x}_0) \leq \alpha & \\ & \vdots & \\ & \mathbf{u}_\tau^p(\mathbf{h} - [E]\mathbf{x}_0) \leq \alpha & \end{array}\right. \tag{B.4}$$

Onde, α é um escalar real. Desde que $\alpha \geq \mathbf{u}_\tau^k(\mathbf{h} - [E]\mathbf{x}_0)$ e o problema é de minimizar a função objetivo α, o valor ótimo será alcançado na igualdade [23].

O problema (B.4) está escrito em termos de $\mathbf{x}_0$ e da variável α. Desde que $\mathbf{x}_0$ seja um ponto adequado implica que $w(\mathbf{x}_0) = 0$, então é possível concluir que o espaço solução de $\mathbf{x}_0$ é restrito pelo seguinte conjunto:

$$\left\{\begin{array}{l} \mathbf{u}_\tau^1(\mathbf{h} - [E]\mathbf{x}_0) \leq 0 \\ \mathbf{u}_\tau^2(\mathbf{h} - [E]\mathbf{x}_0) \leq 0 \\ \vdots \\ \mathbf{u}_\tau^p(\mathbf{h} - [E]\mathbf{x}_0) \leq 0 \end{array}\right.$$

Diante do exposto, a solução do problema (B.1) pode ser alcançada percorrendo-se os passos do seguinte algoritmo.

Algoritmo de Decomposição de Benders

1. Resolver o problema relaxado inicial e ajustar $\mathcal{T}^0 = \emptyset$ e $k = 1$:

$$\left\{\begin{array}{lll} z^k = \text{Min} & \mathbf{c}\mathbf{x}_0 & \\ \mathbf{x}_0 \in \mathcal{X}_0 & & \\ \text{s.a.} & [A]\mathbf{x}_0 \geq \mathbf{b} & \end{array}\right.$$

2. Com o resultado do problema relaxado, $\mathbf{x}_0^k$, resolver em $\mathbf{x}$ o subproblema (B.3) reescrito a seguir:

$$\begin{cases} w(\mathbf{x}_0^k) = \text{Min} & \sum_j s_j \\ \text{s.a.} & [F]\mathbf{x} + [I]\mathbf{s} \geq \mathbf{h} - [E]\mathbf{x}_0^k \\ & \mathbf{s} \geq \mathbf{0} \\ & \mathbf{x}_0^k \quad \text{conhecido} \end{cases}$$

3. Da solução do passo anterior, obtêm-se o objetivo $w^*(\mathbf{x}_0^k)$, o vetor $\mathbf{x}^k$ e o vetor multiplicador $\mathbf{u}_\tau^k$ (solução do dual). Se o valor da função objetivo, $w^*(\mathbf{x}_0^k)$, for igual a zero, os pontos $\mathbf{x}_0^k$ e $\mathbf{x}^k$ serão a solução ótima do problema (B.1), então pare. Em caso contrário, fazer $\mathcal{T}^k \leftarrow \mathcal{T}^{k-1} \cup \{\mathbf{u}_\tau^k\}$ e a partir dos multiplicadores $\mathbf{u}_\tau^1$, $\mathbf{u}_\tau^2$, $\cdots$, $\mathbf{u}_\tau^k$ construir restrições lineares da forma $\mathbf{u}_\tau(\mathbf{h} - [E]\mathbf{x}_0) \leq 0, \quad \forall\, \mathbf{u}_\tau \in \mathcal{T}^k$

4. Com as restrições obtidas no passo anterior formar o problema relaxado

$$\begin{cases} z^k = \text{Min} & \mathbf{c}\mathbf{x}_0 \\ \mathbf{x}_0 \in \mathcal{X}_0 & \\ \text{s.a.} & [A]\mathbf{x}_0 \geq \mathbf{b} \\ & \mathbf{u}_\tau^1(\mathbf{h} - [E]\mathbf{x}_0) \leq 0 \\ & \mathbf{u}_\tau^2(\mathbf{h} - [E]\mathbf{x}_0) \leq 0 \\ & \vdots \\ & \mathbf{u}_\tau^k(\mathbf{h} - [E]\mathbf{x}_0) \leq 0 \end{cases}$$

Resolver o problema relaxado e ajustar $k \leftarrow k + 1$. Retornar ao passo 2.

Como são tomados subconjuntos de $\mathcal{T}$, o problema mostrado no passo 4 é chamado de problema mestre relaxado. O processo iterativo de Benders leva à solução do problema (B.1).

Isto posto, de posse do subproblema primal (ou do seu dual), e em vista da possibilidade de, a partir dos resultados do dual, construir restrições para o problema mestre, a técnica de Benders consiste num processo iterativo entre o problema mestre relaxado e o subproblema dual. A Figura B.1 ilustra a grosso modo a relação entre os dois problemas.

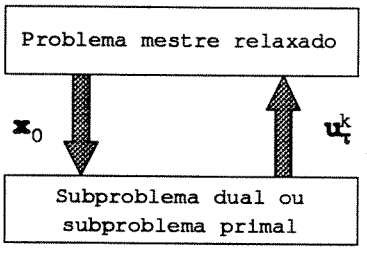


Figura B.1: Técnica de decomposição de Benders.

Apêndice C

Primitivas do PVM3

Neste apêndice é feita uma breve descrição das rotinas usadas com mais freqüência da biblioteca do *PVM3* (**libfpvm3.a**); a finalidade é facilitar o entendimento do conteúdo específico de programação distribuída exposto neste trabalho. Esta seção é útil para uma rápida leitura com vistas a adquirir as primeiras noções sobre as interfaces com o usuário propiciadas pelo *PVM*. Para obter mais informações, uma consulta à referência [52] é indispensável; a referência [53] traz informações sobre aplicações.

call PVMFMYTID(*mytid* **)**
Envolve o processo corrente dentro do *PVM* e gera um único *tid* (= *mytid*) se o processo não foi alocado através da chamada de **PVMFSPAWN()**.

call PVMFEXIT(*info* **)**
Esta rotina informa ao *daemon* local que o processo está deixando o *PVM*.

call PVMFSPAWN(*task, flag, where, ntask, tids, numt* **)**
Supondo-se a existência prévia de um arquivo executável cujo nome é dado pela *string task*, a chamada da rotina **PVMFSPAWN()** aloca na máquina virtual *ntask* cópias deste executável.

call PVMFKILL(*tid, info* **)**
Esta rotina elimina alguma outra tarefa *PVM* identificada pelo *tid*.

call PVMFPARENT(*mtid* **)**
A rotina **PVMFPARENT()** retorna o *tid* (= *mtid* = *master's tid*) do processo que alocou a tarefa corrente.

call PVMFPSTAT(*tid, status* **)**
Esta rotina fornece o *status* de uma tarefa *PVM* identificada pelo *tid*.

call PVMFCONFIG(*nhost, narch, dtid, name, arch, speed, info*)
Esta rotina retorna informações sobre a máquina virtual.

call PVMFTASKS(*which, ntask, tid, ptid, dtid, flag, aout, info*)
Esta rotina retorna informações sobre tarefas *PVM* que estejam executando na máquina virtual.

call PVMFTIDTOHOST(*tid, dtid*)
Se o usuário precisa saber em qual *host* uma tarefa (de *tid* especificado) está executando, então PVMFTIDTOHOST() pode fornecer esta informação.

call PVMFADDHOST(*host, info*)
call PVMFDELHOST(*host, info*)
Estas rotinas respectivamente adiciona e subtrai um membro da máquina virtual.

call PVMFSENDSIG(*tid, signum, info*)
Envia um sinal a outro processo *PVM* identificado pelo *tid*.

call PVMFNOTIFY(*what, msgtag, cnt, tids, info*)
Solicita ao *PVM* para notificar ao processo que a está chamando sobre a detecção de certos eventos, por exemplo, a falha de um *host*.

call PVMFSETOPT(*what, val, oldval*)
call PVMFGETOPT(*what, val*)
São rotinas de propósito geral para permitir ao usuário ajustar ou mostrar opções no sistema *PVM*.

call PVMFBARRIER(*group, count, info*)
Bloqueia o processo que a chamou até que todos os processos integrantes do grupo *group* também a tenha chamado.

call PVMFJOINGROUP(*group, inum*)
Esta rotina insere o processo que a chamou em um grupo especificado por *group*.

call PVMFLVGROUP(*group, info*)
Esta rotina exclui o processo que a chamou de um determinado grupo de nome *group*.

O envio de uma mensagem em *PVM* consiste de três etapas:

(1) inicializar o *buffer*;

(2) a mensagem deve ser "empacotada" dentro do *buffer*;

(3) a mensagem com um todo é enviada para outro processo.

A mensagem ao ser recebida pelo processo destino deve ser "desempacotada" item por item a partir do *buffer* receptor. Dentre as várias opções de transferência de mensagem destacam-se:

call PVMFINITSEND(*encoding*, *bufid*)
Limpa o *buffer* emissor e cria um novo para "empacotar" uma nova mensagem.

call PVMFPACK(*what*, *xp*, *nitem*, *stride*, *info*) ⟶ no processo fonte
call PVMFUNPACK(*what*, *xp*, *nitem*, *stride*, *info*) ⟶ no processo destino.
Cada escalar, vetor, matriz ou *string* para ser transmitido precisa ser "empacotado" no processo fonte e "desempacotado" no respectivo processo destino. Estas são, respectivamente, as funções de PVMFPACK() e PVMFUNPACK().

call PVMFSEND(*tid_dest*, *msgtag*, *info*)
Esta rotina rotula a mensagem com um inteiro *msgtag* e a envia imediatamente para o processo cujo *tid* especificado é *tid_dest*.

call PVMFBCAST(*group*, *msgtag*, *info*)
Rotula a mensagem com um inteiro *msgtag* e difunde a mensagem para todas as tarefas que compõem o grupo *group*.

call PVMFMCAST(*ntask*, *tids*, *msgtag*, *info*)
Rotula a mensagem com um inteiro *msgtag* e envia a mensagem para todas as tarefas especificadas no vetor *tids*.

call PVMFRECV(*tid_source*, *msgtag*, *bufid*)
Esta rotina, ao ser chamada numa aplicação *PVM*, bloqueia o código e espera até que uma mensagem com rótulo *msgtag* tenha chegado proveniente do processo cujo *tid* especificado é *tid_source*.

PVMFNRECV(*tid_source*, *msgtag*, *bufid*)
Verifica se a mensagem requerida chegou sem bloquear o código, podendo ser chamada múltiplas vezes para a mesma mensagem.

Sites na *Internet* sobre *PVM* e Computação Paralela

Informações sobre *PVM*
http://www.epm.ornl.gov/pvm/

Centro Nacional de Processamento de Alto Desempenho em São Paulo (CENAPAD-SP)
http://patmos.cna.unicamp.br/CENAPAD/

Centro de Computação de Alto Desempenho de Cornell (*Cornell Theory Center*)
http://www.tc.cornell.edu/Research/tech.rep.html

Centro de Computação Paralela de Edinburgo (*EPCC*)
http://www.epcc.ed.ac.uk/epcc-tec/documents/

Computação Paralela na França
http://www.irisa.fr/orap

Apêndice D

Um Exemplo Numérico de FPORS via Benders

Neste apêndice, o FPORS do sistema elétrico de 2 barramentos, 2 ramos, 2 geradores e 1 carga, extraído da referência [23], é resolvido passo a passo através da decomposição de Benders.

O objetivo é mostrar o mecanismo de funcionamento do algoritmo no qual se baseia o programa FPORS-III. Cada PL é explicitamente montado neste exemplo, embora isto não seja feito de fato no programa.

A rede-exemplo analisada neste apêndice é a mesma que está ilustrada na Figura 2.2 (Capítulo 2). Neste exemplo numérico, em particular, é considerado o cálculo do fluxo de potência ótimo com restrições de segurança com capacidade de remanejamento pós-contingência. A sistemática usada é a seguinte: monta-se o PL de investimento[1] ou de operação e apresenta-se o resultado correspondente obtido com o programa.

D.1 Rede-exemplo e Dados

Os dados para o programa FPORS-III são dispostos em dois arquivos, conforme mostra a Figura D.1. A rede e os dados operacionais estão mostrados na Figura D.2.

[1] Entenda-se como PL do caso-base.

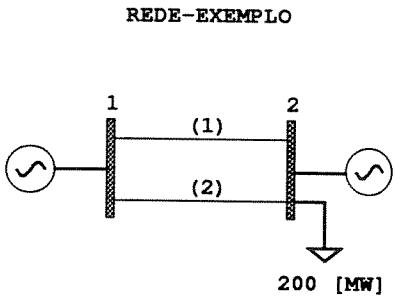
```
   1 1 2 BARRA UM     1000  0. 100.   .0-999. 999.       .0   .0   .0
   2 1 1 BARRA DOIS   1000  0. 100.   .0-999. 999.    200.0   .0   .0
9999
   1       2        .0 100.    .0                          100 100
   1       2        .0 100.    .0                          120 120
9999
   1     2
   1     1
9999
   1    1 R    50.00   200.00   0.000000   1.000000   0.000000    40.00
   1    2 F -9999.00     0.00   0.000000  -4.000000   0.000000     0.00
   2    3 R     0.00   120.00   0.000000   2.000000   0.000000    35.00
   2    4 F     0.00  9999.00   0.000000   4.000000   0.000000     0.00
9999
```

(a)

```
B                vr      ( versao de matriz B'/B'' )
1.0              fres    ( fator para resistencias )
1.0              frea    ( fator para reatancias )
1.00             fpca    ( fator para carga ativa )
1.00             fpger   ( fator para geracao ativa)
1.1111           ffnor   ( fator para limites de fluxos ativos )
1.0              fcont   ( fator para contingencias; 1.0 ==> retira todo o ramo )
1                kord    ( tipo de ordenacao para esparsidade [0 = MD; 1 = MD-ML] )
2                ncorte  ( quantidade de corte para contingencias )
0.1              corte   ( indice severidade de corte para contingencias )
1                jbet    ( fator beta [ 0 = calculado; 1 = unitario ] )
E                obj     ( objetivo FPO-CB [E = economico; D = mdq] )
10.0             alfa    ( fator de custo de racionamento de pot. ATIVA )
2                nseg    ( numero de segmentos da curva de custos [ max 20 ] )
0                kformat ( tipo de formato [0 = PECO; 1 = CDF ] )
DC               flow    ( fluxo de carga inicial AC ou DC - PONTO INICIAL )
0.0001           tol     ( tolerancia convergencia fluxo inicial AC )
0.000001         toler1  ( tolerancia convergencia FPORS [ Sobrecarga ] )
0.000001         toler2  ( tolerancia convergencia FPORS [ Infactibilidade ] )
0.99             fcj     ( fator de obtencao conjunto monitorado-->PL Sub.Operacao )
1.0              fat     ( fator multiplicativo sobre vetor DELTA() )
0                modelo  ( MODO DE PROCESSAMENTO: SINCRONO-->[0] ASSINCRONO-->[1/2] )
```

(b)

Figura D.1: (a) Banco de dados no formato PECO com os dados do fluxo de potência, análise de contingências e dados das unidades de geração. (b) Dados complementares (fatores, parâmetros e opções em geral).



DADOS				
	GERADOR		LINHA	
	1	2	(1)	(2)
$P_g^{mín}$	50	0	P_{km}^{lim}	100 120
$P_g^{máx}$	200	120		
[$/MW]	1	2	Unidades de potência ativa em [MW]	

REMANEJAMENTO PERMITIDO
$\Delta = (40\ 35)^t$ [MW]

Figura D.2: Rede-exemplo com os dados operacionais.

D.2 Solução

(1) Obtenção do ponto inicial de operação através de um fluxo de potência DC.

E S T A D O I N I C I A L DC D O S I S T E M A

BARRAMENTO		TENSAO		GERACAO	CARGA	INJ. LIQ.
No.	EXT. TIPO	MODULO	ANGULO	[MW]	[MW]	[MW]
1	2	1.000	0.000	100.00	0.00	100.00
2	1	1.000	-28.648	100.00	200.00	-100.00

CARGA TOTAL DO SISTEMA [MW] = 200.00

No. BARRAS: 2 No. GERADORES: 4 No. GERAD. FICTI.: 2 No. RAMOS: 2

Figura D.3: Estado inicial do sistema.

Portanto, a geração inicial é a seguinte:

$$\mathbf{P}^{\text{inicial}} = (\ 100\ \ 0\ \ 100\ \ 0\ )^t \quad [MW]$$

As posições 2 e 4 são dos geradores fictícios.

F L U X O I N I C I A L DC D O S I S T E M A
L I M I T E S D E F L U X O D O S I S T E M A

BARRAS		FLUXOS	LIMITES	BARRAS		FLUXOS	LIMITES
INI.	FIM	P(MW)	LIM(MW)	INI.	FIM	P(MW)	LIM(MW)
1	2	50.000	100.000	2	1	-50.000	-100.000
1	2	50.000	120.000	2	1	-50.000	-120.000

No. BARRAS: 2 No. GERADORES: 4 No. GER. FICTI.: 2 No. RAMOS: 2

Figura D.4: Fluxos de potência correspondentes ao ponto inicial.

(2) Forma incremental dos limites das gerações e dos fluxos (em $[pu]$).

$$-0,5 \leq \Delta P_1 \leq 1 \qquad -1,5 \leq \Delta f_{12}^{(1)} \leq 0,5$$

$$-99,99 \leq \Delta P_2 \leq 0 \qquad -1,7 \leq \Delta f_{12}^{(2)} \leq 0,7$$

$$-1 \leq \Delta P_3 \leq 0,2$$

$$0 \leq \Delta P_4 \leq 99,99$$

(3) Solução do FPO caso-base (rede intacta) – 1º PL.

$$\begin{cases} z^1 = \text{Min} \ (1 \quad -40 \quad 2 \quad 40)\,\mathbf{\Delta P} \\ \text{s.a.} \qquad \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 = 0 \\ \qquad\quad -1,5 \leq -0,5\Delta P_3 - 0,5\Delta P_4 \leq 0,5 \\ \qquad\quad -1,7 \leq -0,5\Delta P_3 - 0,5\Delta P_4 \leq 0,7 \\ \qquad\quad -0,5 \leq \Delta P_1 \leq 1 \\ \qquad\quad -99,99 \leq \Delta P_2 \leq 0 \\ \qquad\quad -1 \leq \Delta P_3 \leq 0,2 \\ \qquad\quad 0 \leq \Delta P_4 \leq 99,99 \end{cases}$$

A resposta para este PL é a seguinte:

$$\text{Incremental} \quad \Delta\mathbf{P}^{0,*} = (\,1 \;\; 0 \;\; -1 \;\; 0\,)^t$$

$$\text{Não-incremental} \quad \mathbf{P}^{0,*} = (\,200 \;\; 0 \;\; 0 \;\; 0\,)^t \quad [MW]$$

$$\text{Custo} = 200 \quad [\$]$$

O PL resolvido no passo 3 é o 1º problema de investimento, onde foram relaxadas todas as restrições de segurança. Nos PLs de investimento, onde está $\Delta\mathbf{P}$ entenda-se $\Delta\mathbf{P}^0$.

(4) Classificação das sobrecargas para as contingências das linhas (1) e (2).

R A N K I N G D E S E V E R I D A D E
D E C O N T I N G E N C I A S - SOBRECARGAS

SEVERIDADE	RAMO	TERMINAIS		INDICE	SOBREC
ORDEM	No.	NI	NF	PIMW	[PU]
1	2	1	2	4.00008	1.00001
2	1	1	2	2.77783	0.80001

LISTA INICIAL = 2 LISTA REDUZIDA = 2
INDICE -1. INDICA ILHAMENTO No. ILHAMENTOS = 0

Figura D.5: Classificação das contingências quanto à severidade das sobrecargas.

Para tornar mais simples a solução deste exemplo particular, será considerada apenas a contingência do ramo (2). Ressalta-se que este tipo de simplificação não é feita nos testes dos programas.

(5) Solução do subproblema de operação 1 (rede alterada – sem o ramo (2)). Nos PLs dos subproblemas de operação, onde se encontra o símbolo $\Delta\mathbf{P}$ entenda-se $\Delta\mathbf{P}^{\nu}$, sendo ν o número da contingência.

$$\begin{cases} w_1^1 = \text{Min} \quad (0 \quad -1 \quad 0 \quad 1)\,\mathbf{\Delta P} \\ \text{s.a.} \qquad \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 = 0 \\ \qquad\qquad -3 \leq -\Delta P_3 - \Delta P_4 \leq -1 \\ \qquad\qquad -0,4 \leq \Delta P_1 \leq 0 \\ \qquad\qquad -99,99 \leq \Delta P_2 \leq 0 \\ \qquad\qquad 0 \leq \Delta P_3 \leq 0,35 \\ \qquad\qquad 0 \leq \Delta P_4 \leq 99,99 \end{cases}$$

A resposta para este PL é a seguinte:

$$\text{Incremental} \quad \mathbf{\Delta P}^{1,*} = (-0,4 \quad -0,6 \quad 0,35 \quad 0,65)^t$$

$$w_1^{1,*} = 1,25 \rightarrow \text{medida da "infactibilidade"}$$

O corte de Benders é (vide equação (2.17)):

$$\Delta P_1 - \Delta P_2 - \Delta P_3 - 2\Delta P_4 \leq 0,74998$$

(6) Solução do 2º problema de investimento (rede intacta com o corte de Benders).

$$\begin{cases} z^2 = \text{Min} \quad (1 \quad -40 \quad 2 \quad 40)\,\mathbf{\Delta P} \\ \text{s.a.} \qquad \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 = 0 \\ \qquad\qquad -1,5 \leq -0,5\Delta P_3 - 0,5\Delta P_4 \leq 0,5 \\ \qquad\qquad -1,7 \leq -0,5\Delta P_3 - 0,5\Delta P_4 \leq 0,7 \\ \qquad\qquad -0,5 \leq \Delta P_1 \leq 1 \\ \qquad\qquad -99,99 \leq \Delta P_2 \leq 0 \\ \qquad\qquad -1 \leq \Delta P_3 \leq 0,2 \\ \qquad\qquad 0 \leq \Delta P_4 \leq 99,99 \\ \qquad\qquad \Delta P_1 - \Delta P_2 - \Delta P_3 - 2\Delta P_4 \leq 0,74998 \leftarrow \text{corte de Benders} \end{cases}$$

A resposta para este PL é a seguinte:

$$\text{Incremental} \quad \mathbf{\Delta P}^{0,*} = (\, 0,37499 \;\; 0 \;\; -0,37499 \;\; 0\,)^t$$

(7) Solução do subproblema de operação 2 (rede alterada – sem o ramo (2)).

$$\begin{cases} w_2^1 = \text{Min} \;\; (\, 0 \;\; -1 \;\; 0 \;\; 1\,)\mathbf{\Delta P} \\ \text{s.a.} \qquad \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 = 0 \\ \qquad\qquad -2,375 \leq -\Delta P_3 - \Delta P_4 \leq -0,375 \\ \qquad\qquad -0,4 \leq \Delta P_1 \leq 0,4 \\ \qquad\qquad -99,99 \leq \Delta P_2 \leq 0 \\ \qquad\qquad -0,35 \leq \Delta P_3 \leq 0,35 \\ \qquad\qquad 0 \leq \Delta P_4 \leq 99,99 \end{cases}$$

A resposta para este PL é a seguinte:

$$\text{Incremental} \quad \mathbf{\Delta P}^{1,*} = (\, -0,375 \;\; 0 \;\; 0,375 \;\; 0,025\,)^t$$

$$w_2^{1,*} = 0,025 \rightarrow \text{medida da "infactibilidade"}$$

O corte de Benders é:

$$-\Delta P_2 - \Delta P_3 - \Delta P_4 \leq 0,34999$$

(8) Solução do 3º problema de investimento (rede intacta com os cortes de Benders).

$$
\begin{cases}
z^3 = \text{Min} \quad (\,1 \quad -40 \quad 2 \quad 40\,)\,\mathbf{\Delta P} \\
\text{s.a.} \qquad \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 = 0 \\
\qquad\qquad -1{,}5 \leq -0{,}5\Delta P_3 - 0{,}5\Delta P_4 \leq 0{,}5 \\
\qquad\qquad -1{,}7 \leq -0{,}5\Delta P_3 - 0{,}5\Delta P_4 \leq 0{,}7 \\
\qquad\qquad -0{,}5 \leq \Delta P_1 \leq 1 \\
\qquad\qquad -99{,}99 \leq \Delta P_2 \leq 0 \\
\qquad\qquad -1 \leq \Delta P_3 \leq 0{,}2 \\
\qquad\qquad 0 \leq \Delta P_4 \leq 99{,}99 \\
\qquad\qquad \Delta P_1 - \Delta P_2 - \Delta P_3 - 2\Delta P_4 \leq 0{,}74998 \;\leftarrow\; 1^{\text{o}}\text{corte de Benders} \\
\qquad\qquad -\Delta P_2 - \Delta P_3 - \Delta P_4 \leq 0{,}34999 \;\leftarrow\; 2^{\text{o}}\text{corte de Benders}
\end{cases}
$$

A resposta para este PL é a seguinte:

$$\text{Incremental} \quad \mathbf{\Delta P}^{0,*} = (\,0{,}34999 \quad 0 \quad -0{,}34999 \quad 0\,)^t$$

$$\text{Não-incremental} \quad \mathbf{P}^{0,*} = (\,135 \quad 0 \quad 65 \quad 0\,)^t \quad [MW]$$

$$\text{Custo} = 265 \quad [\$]$$

(9) Solução do subproblema de operação 3 (rede alterada – sem o ramo (2)).

$$\begin{cases} w_3^1 = \text{Min} \quad (0 \quad -1 \quad 0 \quad 1)\,\Delta\mathbf{P} \\ \text{s.a.} \qquad \Delta P_1 + \Delta P_2 + \Delta P_3 + \Delta P_4 = 0 \\ \qquad\qquad -2,35 \leq -\Delta P_3 - \Delta P_4 \leq -0,35 \\ \qquad\qquad -0,4 \leq \Delta P_1 \leq 0,4 \\ \qquad\qquad -99,99 \leq \Delta P_2 \leq 0 \\ \qquad\qquad -0,3 \leq \Delta P_3 \leq 0,35 \\ \qquad\qquad 0 \leq \Delta P_4 \leq 99,99 \end{cases}$$

A resposta para este PL é a seguinte:

$$\text{Incremental} \quad \Delta\mathbf{P}^{1,*} = (-0,35 \quad 0 \quad 0,35 \quad 0)^t$$

$$w_3^{1,*} = 0 \rightarrow \text{"infactibilidade" eliminada}$$

Se a "infactibilidade" no subproblema de operação construído em torno do 3º problema de investimento foi eliminada, significa que a solução ótima-segura é:

$$\text{Incremental} \quad \Delta\mathbf{P}^{0,*} = (0,34999 \quad 0 \quad -0,34999 \quad 0)^t$$

$$\text{Não-incremental} \quad \mathbf{P}^{0,*} = (135 \quad 0 \quad 65 \quad 0)^t \quad [MW]$$

$$\text{Custo} = 265 \quad [\$]$$

Este resultado confere com a saída em arquivo do programa FPORS-III.

Apêndice E

Avaliação Crítica de Métodos Diretos e Iterativos para Resolver Sistemas $[A]\mathbf{x} = \mathbf{b}$ em Cálculos de Fluxo de Potência e Análise de Contingências

CRITICAL EVALUATION OF DIRECT AND ITERATIVE METHODS FOR SOLVING Ax = b SYSTEMS IN POWER FLOW CALCULATIONS AND CONTINGENCY ANALYSIS

A.B. Alves, E.N. Asada and A. Monticelli
Unicamp-Campinas-Brazil

Abstract: Studies comparing the performance of direct and iterative solvers of $A\mathbf{x} = \mathbf{b}$ systems have been performed. By direct methods we mean sparsity preserving Gaussian elimination based approaches, whereas by iterative methods we mean pre-conditioned conjugate gradient methods such as the ones based on incomplete factorization of matrix A (e.g., incomplete Cholesky and K_0 pre-conditioners). A new ordering scheme has been developed which decreases the number of iterations required by iterative methods to reach convergence and an alternative conditioning approach is suggested in connection with the contingency screening problem. Tests have been performed with networks varying from 14 to 1663 buses, including base-case and contingency cases.

Keywords: static contingency analysis, power flow analysis, $A\mathbf{x} = \mathbf{b}$ solvers, sparsity preserving methods, conjugate gradient methods.

INTRODUCTION

It is well known that iterative methods for solving large sparse sets of linear equations, $A\mathbf{x} = \mathbf{b}$, can naturally profit from parallelization and vectorization. The same is not generally true regarding sparsity preserving methods commonly used in power flow calculations: direct methods are known to be hard to parallelize/vectorize. Comparative studies between direct and iterative methods using parallel or vector machines tend to introduce a bias in favor of the iterative methods and makes it difficult to evaluate which part of the benefits is due to the hardware and which part is due to the method itself. Although recognizing that parallelization/vectorization should be used whenever advantageous, in this paper we carried out studies comparing direct and iterative methods using conventional serial machines (without parallelization or vectorization facilities) [1-3].

Efficient solvers for sets of linear equations, $A\mathbf{x} = \mathbf{b}$, are essential in power flow calculation and contingency analysis. This paper specifically deals with contingency screening methods: linear contingency screening with decoupled $P\theta$ and QV half-iterations. A number of different versions of iterative methods have been implemented and tested. Comparative studies have been performed with direct methods based on sparsity preserving techniques: sparse matrix and sparse vector techniques [4-8].

Sparsity preserving ordering schemes (e.g., Tinney-2 scheme) are a crucial part of direct methods. This type of ordering is closely related to complete factorization of the coefficient matrix A and so it is not normally applicable to iterative methods based on incomplete factorizations such as Cholesky's and the K_0 pre-conditioners. Alternative ordering schemes for iterative methods were investigated and tested.

More specifically, a new ordering scheme is proposed for pre-conditioners based on incomplete factorization. In addition, a new pre-conditioner which is naturally suited to contingency analysis is also suggested: the idea is to use the complete factors obtained for the base-case as a pre-conditioner for contingency cases. Such improvements in the iterative methods make the comparisons with the conventional direct methods (based on well established sparse matrix/vector procedures) more meaningful, as is fully illustrated in the paper.

The paper also shows how ordering can affect the eigenvalue spectrum, which in turn has an impact on convergence rates of the pre-conditioned gradient methods. The number of conjugate gradient iterations is almost directly related to the norm of the remainder matrix R ($A = M + R$, where M is the pre-conditioner, A is the coefficient matrix and R is the remainder) [9]. While in direct methods the goal is usually to minimize storage and the number of floating-point operations, for incomplete elimination where no fill-in is allowed, network parameters, rather than matrix structure, are the issue. Although sparsity is preserved, ordering may affect the size of the norm of R. A good ordering scheme is aimed at making $M^{-1}A$ to be close to the identity matrix [9]: ideally the spectrum of $M^{-1}A$ will have a single eigenvalue; in practice we content ourselves with a spectrum containing a small number of clusters of eigenvalues. The clustering of eigenvalues produced by a good pre-conditioner in general also reduces the spectral condition number of the modified coefficient matrix ($\kappa_2(\hat{A}) = \lambda_{max}(\hat{A})/\lambda_{min}(\hat{A})$, where λ_{max} is the maximum and λ_{min} is the minimum eigenvalue of $\hat{A}$) [10,11].

PRE-CONDITIONED CONJUGATE GRADIENT

The pre-conditioned conjugate gradient algorithm is obtained by implicitly transforming the coefficient matrix A into a matrix $\hat{A}$ with a clustered spectrum (which, for the networks we have studied in this paper, usually produces a reduction in the number of distinct eigenvalues and implies a reduction in the spectral condition number, $\kappa(\hat{A}) < \kappa(A)$) as shown in Figures 4 and 7 [10-13]. The linear transformation $\hat{A} = C^{-1}AC^{-\mathrm{T}}$ is applied to the coefficient matrix A in order to derive the PCG method, where C is a nonsingular matrix such that $\kappa((CC^{\mathrm{T}})^{-1}A) < \kappa(A)$. The transformed set of linear equations is $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, where $\hat{\mathbf{x}} = C^{\mathrm{T}}\mathbf{x}$ and $\hat{\mathbf{b}} = C^{-1}\mathbf{b}$. Matrix CC^{T} can be used as a pre-conditioner. In the algorithm given below it is not necessary to compute this matrix explicitly. $M = CC^{\mathrm{T}}$ is a symmetric positive definite matrix.

The PCG algorithm is summarized in the following [3,10]:

1. Begin
2. $\mathbf{x} := \mathbf{0}$; $\mathbf{r} := \mathbf{b}$; $k := 1$
3. While $(max|r_i| > \epsilon)$ and $(k < k_{max})$ do
4. Begin
5. Solve $M\mathbf{z} = \mathbf{r}$
6. $\gamma_{k-1} := \mathbf{r}^T\mathbf{z}$
7. If $(k = 1)$ then $\mathbf{p} := \mathbf{z}$
8. else Begin
9. $\beta_k := \gamma_{k-1}/\gamma_{k-2}$
10. $\mathbf{p} := \mathbf{z} + \beta_k\mathbf{p}$
11. End
12. $\mathbf{w} := A\mathbf{p}$, $\alpha_k := \gamma_{k-1}/\mathbf{p}^T\mathbf{w}$
13. $\mathbf{x} := \mathbf{x} + \alpha_k\mathbf{p}$
14. $\mathbf{r} := \mathbf{r} - \alpha_k\mathbf{w}$, $k := k+1$
15. End *while*
16. End

Incomplete Factorization: Ignoring Fill-ins

The Cholesky factorization of the coefficient matrix A is given by $A = LL^{\mathrm{T}}$, where L is a lower triangular matrix. Let $\tilde{L}$ be an approximation to L obtained by ignoring the fill-in elements normally created during the factorization of A, i.e., $A = \tilde{L}\tilde{L}^{\mathrm{T}} + R$, where $R \neq 0$ is an error matrix [9]. The matrix M which appears in the pre-conditioned conjugate gradient algorithm above is given by $M = \tilde{L}\tilde{L}^{\mathrm{T}}$. As mentioned above, this matrix is not computed explicitly since only its triangular factors $\tilde{L}$ and $\tilde{L}^{\mathrm{T}}$ are needed (Herein, $M = \tilde{L}\tilde{D}\tilde{L}^{\mathrm{T}}$).

The K_0 Pre-conditioner

The pre-conditioning matrix K_0 is obtained by applying Gaussian elimination to the coefficient matrix, A, precluding the creation of fill-in elements and ignoring all the updates in the existing off-diagonal positions [14]. That is to say, only the diagonal elements are updated during this type of incomplete factorization. The resulting K_0 matrix is given by

$$K_0 = (\overline{L} + D_0){D_0}^{-1}(\overline{L}^{\mathrm{T}} + D_0)$$

The triangular matrix $\overline{L}$ contains the original off-diagonal elements of the coefficient matrix A. Matrix K_0 is such that $diag(K_0 - A)$ is equal to zero. This matrix is never explicitly computed since only the elements of D_0 are needed (Notice that $\tilde{L} = \overline{L} + D_0$ and $\tilde{D} = {D_0}^{-1}$).

The Incomplete Cholesky, IC, Pre-conditioner

The square-root free Cholesky algorithm is based on the expressions,

$$\begin{aligned} L_{ji} &= A_{ji} - \textstyle\sum_{k=1}^{i-1} L_{jk}L_{ik}D_{kk} \\ j &= i, (i+1), \cdots, n \\ D_{ii} &= (L_{ii})^{-1} \end{aligned} \qquad (1)$$

A simple yet efficient way of obtaining the incomplete Cholesky factors consists in forcing that all the elements L_{ji} corresponding to fills be equal to zero, i.e., the original sparsity structure of A is kept throughout factorization [15].

Pre-conditioners for Contingency Screening

Three types of pre-conditioners have been tested for contingency screening problem ($1P\theta$-$1QV$ fast decoupled half-iterations): K_0, incomplete Cholesky (IC), and complete Cholesky (CC). The three pre-conditioners use the same PCG algorithm described above, and for all of them the conditioning matrices are kept constant throughout contingency screening: i.e., the conditioning matrices (i.e., the triangular factors) are computed only for the base-case topology and used for all the topologies resulting from contingency cases. Only single/multiple branch contingencies not leading to disconnections or other types of structural changes have been considered in the tests performed.

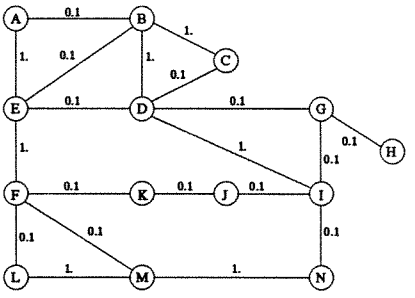


Figure 1: *IEEE-14 network with modified branch reactances used to illustrate the minimum spanning tree algorithm.*

NEW ORDERING SCHEME FOR PRE-CONDITIONERS BASED ON INCOMPLETE FACTORIZATION

In order to make meaningful comparisons between direct and iterative methods it is necessary to make sure we are having the same general testing conditions for both approaches. One critical issue is ordering. In this section we suggest a new ordering scheme which seems to be most adequate for the iterative methods (we believe that this scheme brings to the iterative methods the same type of benefits that Tinney-2 scheme brings to the direct methods). Although in the PCG methods we are considering herein structure is invariant (no fill-ins are considered), it is shown that network parameters may affect convergence. Eigenvalue clustering (and clumping) produced by new ordering scheme is fully discussed as well.

Minimum Spanning Tree Algorithm

The incomplete factorization schemes K_0 and IC preclude the creation of fill-ins in the triangular factors of its pre-conditioning matrices. Thus it might be expected that schemes such as Tinney-2 normally used in connection with direct methods could improve convergence, since the number

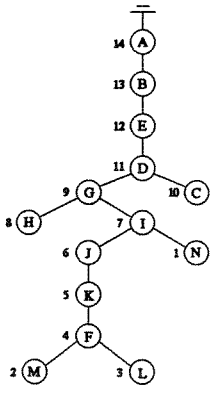


Figure 2: *Minimum Spanning Tree corresponding to the modified IEEE-14 network given in Fig.1.*

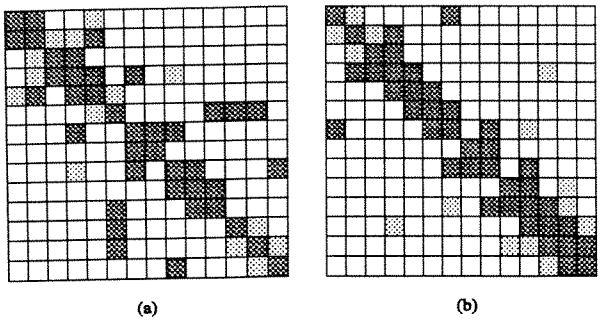


Figure 3: *Effect of the MST ordering on the structure of the B' matrix for the modified version of the IEEE 14-bus network as given in Fig.1: (a) unordered; (b) MST ordering.*

of fills in nearly minimized (and thus the error would be approximately minimized as well). Indeed this can be observed in practice as illustrated in a number of cases shown in the test section. The use of this scheme however is not a practical choice, since it makes sense only if we are interested in the fill-in elements as happens with complete factorization methods. Nevertheless at least for benchmark purposes schemes such as Tinney's can be used. The impact on convergence rates of permuting rows and columns of A was noted in [9,11,13,16] where it is suggested that a diagonally dominant matrix will decrease the number of iterations.

In the following a new scheme for ordering pivot operations in incomplete factorization methods (K_0 and IC preconditioners) is proposed: the *minimum spanning tree* approach. The idea is to obtain a spanning tree containing the strongest links of the network and then number the nodes of the tree in such a way that the leaves of the tree are numbered first; once the leaves are processed they are discarded and the new leaves are numbered until no leaves are left.

The Kruskal algorithm given below, with complexity $O(m \log_2 m)$, is able to obtain the minimum spanning tree (MST). Node ordering is readily obtained from the minimum spanning tree.

1. Begin
2. Given a network with n nodes and m branches. Order the branches in ascending mode according of its costs, $|x_{ij}|$;
3. Select a branch with the minimum cost to enter the tree, **T**:

   If the branch forms a closed loop in **T** then REJECT;

   Otherwise, ACCEPT;

   STOP if the number of accepted branches is equal to $n - 1$.
4. End

Illustrative Example

Figure 1 shows a modified version of the IEEE-14 system in which the branch reactances have been modified in order to emphasize the way the minimum spanning tree algorithm operates. Fig.2 shows the resulting minimum spanning tree. Fig.3 illustrates the effect of the MST ordering on the structure of the B' matrix (higher density closer to the main diagonal, $B'_{ij} \geq 10.$).

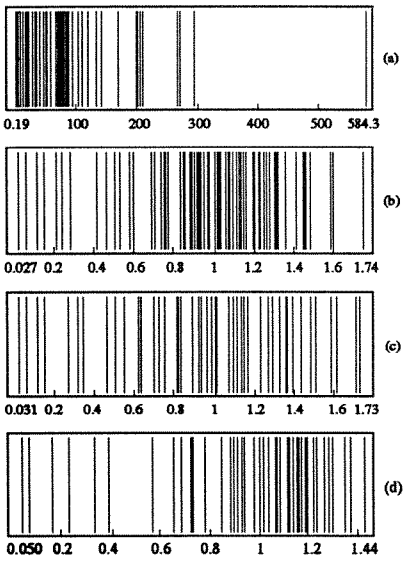


Figure 4: *Impact of pre-conditioning on eigenvalue spectrum for the IEEE-118 network.*

The Impact of Ordering on Eigenvalue Spectrum

In this section we show empirically how ordering can affect eigenvalue spectrum, which in turn has an impact on convergence rates. Herein matrix M is an approximation to matrix A, such that $A = M + R$, where R is a remainder matrix. The number of conjugate gradient iterations is almost directly related to the norm of the error R (It is then suggested that $\|R\|$ can be used to estimate the condition numbers of incomplete factorization methods) [9]. While in direct methods the goal is usually to minimize storage and the number of floating-point operations, for incomplete elimination where no fill-in is allowed, network parameters, rather than matrix structure, are the issue. Although structure is a given (i.e., sparsity is preserved) ordering may affect the size of the norm R and a good ordering scheme will make $M^{-1}A$ to be close to the identity matrix [9].

Table 1 shows two different measures of the remainder matrix R: the Frobenius norm $\left(\sum_{i,j} |R_{ij}|^2\right)^{1/2}$ and a maximum entry in R ($\max|R_{ij}|$), for incomplete Cholesky preconditioner. Both amounts are affected by ordering although the Frobenius norm is more affected than the $\max|R_{ij}|$ since it is a more accurate measurement. Table 2 shows that the spectral condition numbers present a similar behavior.

Table 1: *Effect of ordering on quality of IC pre-conditioning for the most critical single contingency (please refer to spectrum of the IEEE 118-bus network given in Fig.4).*

Network	$\|R\|_F$ ($\max\|R_{ij}\|$)		
	Unordered	Tinney-2	MST
IEEE-30	21.4 (9.1)	20.5 (9.1)	19.1 (9.1)
IEEE-57	51.2 (19.1)	44.1 (19.1)	40.5 (19.1)
IEEE-118	110. (39.5)	102.1 (25.5)	68.4 (25.)

Convergence of conjugate gradient algorithms is determined by the spectrum of the iteration matrix $M^{-1}A$ [9,12]. Figure 4 gives the spectrum obtained with the IEEE 118-bus network for the topology corresponding to the most critical single branch contingency (we are considering the B' matrix as in the BX version of the fast decoupled power flow). Four cases are shown in the figure: (a) spectrum without pre-conditioning; (b) pre-conditioning without ordering; (c)

pre-conditioning with Tinney-2 ordering scheme; and (d) pre-conditioning with MST ordering scheme. The extremal values indicated are λ_{min} and λ_{max}. It can be observed that ordering produces both clustering and clumping the eigenvalue spectrum and these effects are more pronounced with the MST algorithm than with Tinney-2 scheme. As a result, the convergence rates obtained with the PCG algorithm using MST ordering scheme are better than the convergence rates with the Tinney-2 scheme (Table 3).

Table 2: *Effect of ordering on quality of IC pre-conditioning for the most critical single contingency (please refer to spectrum of the IEEE 118-bus network given in Fig.4).*

Network	Spectral condition number $\lambda_{max}/\lambda_{min}$		
	Unordered	Tinney-2	MST
IEEE-30	18.5	13.8	8.3
IEEE-57	22.4	14.1	9.6
IEEE-118	64.4	55.8	28.8

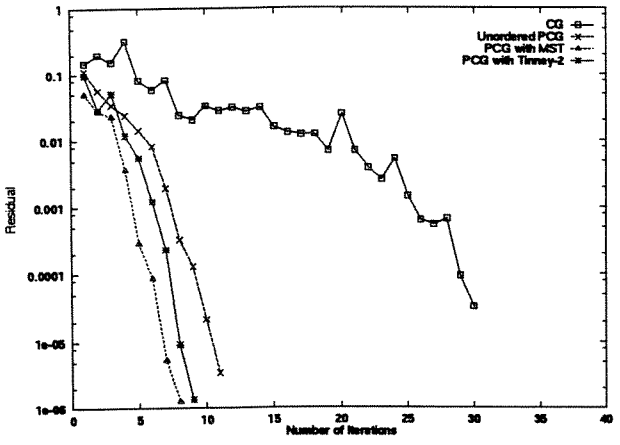


Figure 5: *The impact of the ordering scheme on the convergence rate for the IEEE 30-bus system for the most critical contingency.*

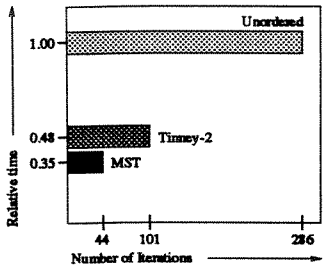


Figure 6: *Impact of ordering on average elapsed times for the most critical single contingency of the BR-1663 network.*

The Impact of Ordering on Convergence Rates

Figure 5 illustrates the effect of the ordering scheme on the convergence rate of the conjugate gradient algorithms for the IEEE 30-bus system. Four versions of the CG method are compared in the figure: (i) conjugate gradient algorithm without pre-conditioning [10]; (ii) unordered pre-conditioned conjugate gradient (i.e., with the arbitrary order with which the buses appear in the input file); (iii) pre-conditioned conjugate gradient with Tinney-2 ordering scheme; and (iv) pre-conditioned conjugate gradient with the minimum spanning tree ordering scheme. As mentioned above, the Tinney-2 scheme, in general, is not a practical choice for iterative methods and it is shown here only for the sake of comparison, since this method somehow minimizes the number of fills and so indirectly should reduce the impact of neglecting the fill-in elements as happens with the incomplete factorization.

Finally Table 3 shows number of iterations which as expected also demonstrates the benefits brought by Tinney-2 and MST ordering schemes. The same type of behavior have been observed with larger networks as shown in the test results section. As a matter of fact, improvements in convergence shown by the practical systems tested are even more pronounced.

Table 3: *Effect of ordering on PCG performance for the most critical single contingency (tolerance 5.10^{-6}), using the IC pre-conditioner (please see spectrum in Fig.4 for the IEEE 118-bus network).*

Network	Number of iterations		
	Unordered	Tinney-2	MST
IEEE-30	11	9	8
IEEE-57	16	13	10
IEEE-118	24	22	18

The Impact of Ordering on Computing Times

The normalized elapsed times, considering three different orderings, for the most critical single contingency of the BR-1663 network (branch 166) are illustrated in Figure 6.

A PRE-CONDITIONER FOR CONTINGENCY ANALYSIS

Motivation

If we could find a pre-conditioner such that all the eigenvalues of $M^{-1}A$ were equal, then the PCG method would converge in one iteration [17]. In practice, at least for the contingency analysis problem, we can get close to such ideal situation: the idea is to use the complete factors obtained for the base-case as the pre-conditioner for contingency cases. To illustrate why this can provide good results Figure 7 shows three different spectra: (a) the spectrum corresponding to the base-case B' (BX version) for the IEEE 57-bus network; (b) the spectrum corresponding to the most critical single contingency for the same network; and (c) the spectrum corresponding to $M^{-1}A$, where M is the base-case B' matrix and A the B' matrix corresponding to the topology resulting when the most critical contingency is simulated. Notice that in (c) there are only two eigenvalues in the spectrum: .1631 and 1.000.

Theoretical Justification

Although accurate predictions of the convergence of iterative methods are difficult to make, useful bounds can often be obtained [18]. Ref. [12,17] suggest that the number of independent eigenvalues of the transformed coefficient matrix can be used as an upper-bound for the number of iteration required by the PCG method. Thus in the present example such upper-bound would be two iterations, which explains the results reported in Tables 5-9 for the proposed method (only one iteration is required in most of the contingencies). This result shows that iterative methods may be competitive (with respect to direct methods) even for networks as small as the IEEE 57-bus system.

According to [18], if the extremal eigenvalues of the matrix $M^{-1}A$ are well separated, as in the example given in Fig.7,

then one often observes that convergence rate increases per iteration (*superlinear convergence*). This is due to the fact that CG tends to act primarily on the errors in the direction of eigenvectors associated with extremal eigenvalues. Thus, as convergence progress, the spectral radius of the transformed matrix being processed is reduced, tending to speed-up convergence. In the example system mentioned above, for most contingencies, errors as small as 10^{-14} have been observed after the first iteration of the PCG method with the proposed pre-conditioning.

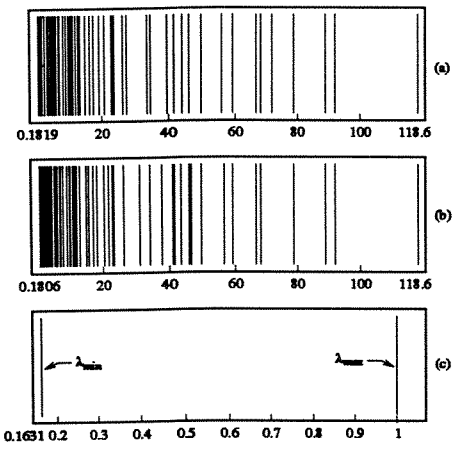


Figure 7: *(a) Spectrum for the IEEE 57-bus network, base-case; (b) most critical single contingency case; (c) spectrum of matrix $M^{-1}A$ (M is the coefficient matrix of the base-case and A is the coefficient matrix of the contingency case).*

DIRECT METHODS

The objective of contingency selection is to identify severe contingencies from a list of possible contingencies. In this application contingency selection is performed with a linearized power flow model (incremental power flow model with sparse vector techniques, $1P\theta$ or $1P\theta$-$1QV$), also called screening. The Tinney-2 ordering scheme is used for preserving matrix sparsity. Severe contingencies are ranked according to performance indices. The direct method is based on Gaussian elimination procedures applied to large sparse matrices. Screening in the direct method use compensation techniques (Matrix Modification Lemma) or sparse matrix factor updating are used according to the type of contingency under analysis (single, multiple) [4-8].

TEST RESULTS

Table 4 summarizes the main characteristics of the power networks used in the tests. The BR-810 and BR-1663 networks represent parts of the Brazilian Southern-Southeastern power grids (up to 750 kV); the other networks are well known IEEE test systems. Single and multiple contingency lists are treated separately to emphasize the effect of the contingency type on the performance of the various methods tested.

Critical contingencies tend to demand more from the iterative methods. Table 5 gives the convergence behavior for the two most critical contingencies (two single (S) and two multiple (M) contingencies) using the pre-conditioned gradient approach with the complete Cholesky (CC) pre-conditioner. The pre-conditioning matrices (triangular factors) are obtained for the base-case $B'(B'')$ matrix and reused for contingency analysis. The convergence tolerance used in these cases was 5.10^{-6}. It is interesting to notice that the Tinney-2 ordering scheme consistently produces performances which are in between the performances obtained with the minimum spanning tree scheme and the unordered scheme. Table 6 shows the behavior of the CC conditioner for multiple contingencies cases (a selection of hard cases) and with convergence tolerance of 5.10^{-6} (the actual accuracies obtained are much higher however: typically 10^{-10} in these cases).

Table 4: *Test network characteristics.*

Networks with contingencies	Number of buses	Number of branches	Number of contingencies
IEEE-30	30	41	30 single
IEEE-30*	30	41	30 multiple
IEEE-57	57	80	30 single
IEEE-57*	57	80	30 multiple
IEEE-118	118	179	100 single
IEEE-118*	118	179	100 multiple
IEEE-300	300	411	100 single
IEEE-300*	300	411	100 multiple
BR-810	810	1340	100 single
BR-810*	810	1340	100 multiple
BR-1663	1663	2349	100 single
BR-1663*	1663	2349	100 multiple

Table 5: *Number of iterations for the IC and CC pre-conditioners for the two most critical contingencies in the list (Single (S) and Multiple (M)), convergence tolerance of 5.10^{-6}. All pre-conditioning matrices (triangular factors) are obtained for the base-case $B'(B'')$ matrix and reused for contingency screening; test network is the BR-1663.*

Screening	Pre-conditioners and ordering schemes			
	IC			CC
	Unordered	Tinney-2	MST	Tinney-2
S ($1P\theta$)	286	101	44	1
S ($1P\theta$)	243	83	38	1
M ($1P\theta$)	320	110	48	7
M ($1P\theta$)	292	103	49	3
S ($1P\theta$-QV)	286/220	101/67	44/28	1/2
S ($1P\theta$-QV)	243/146	83/45	38/16	1/2
M ($1P\theta$-QV)	320/266	110/68	48/35	7/7
M ($1P\theta$-QV)	292/247	103/66	49/29	3/4

Table 6: *Number of iterations (minimum-average-maximum) for the CC pre-conditioner; multiple contingencies; convergence tolerance of 5.10^{-6} (Elapsed times for the $1P\theta$ cases are given in Tables 8 and 9).*

Network	Screening	
	$1P\theta$ min-avg-max	$1P\theta$-QV min-avg-max
IEEE-30*	2-3-7	2-3-7/1-3-6
IEEE-57*	2-3-7	2-3-7/1-2-5
IEEE-118*	2-3-7	2-3-7/1-2-6
IEEE-300*	2-3-7	2-3-7/2-3-7
BR-810*	1-3-6	1-3-6/1-3-7
BR-1663*	1-3-7	1-3-7/2-3-7

Table 7: *Number of iterations (minimum-average-maximum) for pre-conditioners K_0, IC and CC; $1P\theta$ screening, convergence tolerance of 5.10^{-6}, single contingencies. All the pre-conditioning matrices (triangular factors) are obtained for the base-case B' matrix and reused for all contingencies in the list.*

Network	$\sqrt{n}$	Pre-conditioners and ordering schemes						
		K_0			IC			CC
		Unordered	Tinney-2	MST	Unordered	Tinney-2	MST	Tinney-2
IEEE-30	5.48	9-11-12	8-10-11	6-8-10	8-11-12	7-9-10	6-7-8	1-1-1
IEEE-57	7.55	10-14-16	9-12-14	7-9-11	11-14-16	8-11-13	6-9-10	1-1-1
IEEE-118	10.86	14-22-27	10-21-26	11-16-19	13-21-26	12-20-24	9-15-18	1-1-1
IEEE-300	17.32	36-46-53	1-39-43	1-25-30	37-45-53	1-34-41	1-22-27	1-1-1
BR-810	28.46	1-85-119	1-53-74	1-25-35	1-84-117	1-46-74	1-21-32	1-1-1
BR-1663	40.78	27-225-375	1-78-101	1-37-53	27-218-286	1-74-101	1-32-44	1-1-1

Table 8: *Elapsed times in seconds for base-case solution and contingency screening $1P\theta$; comparisons between the direct method and the pre-conditioned conjugate gradient method with complete Cholesky base-case triangular factors used for pre-conditioning (tolerance of 5.10^{-6}); IBM-RISC/6000 machine; single contingencies; multiple contingencies.*

Network	PCG with CC	Direct	Difference
IEEE-30	0.063	0.061	+3.2 %
IEEE-57	0.121	0.123	-1.7 %
IEEE-118	0.257	0.237	+7.8 %
IEEE-300	0.485	0.422	+13. %
BR-810	1.517	1.379	+9.1 %
BR-1663	3.264	2.619	+19.8 %
Network	PCG with CC	Direct	Difference
IEEE-30*	0.072	0.081	-12.5 %
IEEE-57*	0.094	0.103	-9.6 %
IEEE-118*	0.367	0.411	-12. %
IEEE-300*	0.826	0.735	+11. %
BR-810*	2.776	2.820	-1.6 %
BR-1663*	5.866	5.071	+13.6 %

Table 9: *Elapsed times in seconds for base-case solution and contingency screening $1P\theta$; comparisons between the direct method and the pre-conditioned conjugate gradient method with complete Cholesky base-case triangular factors used for pre-conditioning (tolerance of 5.10^{-6}); SUN SPARCstation 2 machine; single contingencies; multiple contingencies.*

Network	PCG with CC	Direct	Difference
IEEE-30	0.274	0.242	+11.7 %
IEEE-57	0.327	0.526	-60.9 %
IEEE-118	1.317	1.515	-15. %
IEEE-300	3.117	2.860	+8.2 %
BR-810	10.020	9.490	+5.3 %
BR-1663	20.502	16.431	+19.8 %
Network	PCG with CC	Direct	Difference
IEEE-30*	0.454	0.654	-44. %
IEEE-57*	0.659	1.063	-61.3 %
IEEE-118*	2.727	2.849	-4.5 %
IEEE-300*	6.948	6.013	+13.5 %
BR-810*	19.532	18.172	+7. %
BR-1663*	39.264	29.765	+24.2 %

Table 7 gives the convergence behavior (minimum-average-maximum numbers of iterations) for contingency screening using the pre-conditioners K_0, incomplete Cholesky (IC), and complete Cholesky (CC) for the base-case matrix. As described in the previous section, all pre-conditioning matrices (triangular factors) are obtained for the base-case B' matrix and are reused for all contingencies in the list. The convergence tolerance used in these cases was 5.10^{-6}. Example on how to read the table: for the IC method, Tinney-2 scheme and the IEEE 57-bus network we have the entry 8-11-13 which means that the minimum number of iterations required by a contingency case was 8, the average number of iterations over all the contingencies was 11, and the maximum number of iterations observed was 13.

Computing times (elapsed times in seconds excluding I/O), using *IBM-RISC/6000* machine, model *250 (PowerPC)*, for base-case solution and contingency screening are shown in Table 8 (single/multiple contingency lists). The table compares direct and the pre-conditioned conjugate gradient method with complete Cholesky base-case triangular factors used for pre-conditioning. Notice that the upper part of the table refers to single contingency lists while the lower part refers to multiple contingency lists. Table 9 presents the same type the results now obtained with the *SUN SPARCstation 2*. The iterative method showed to be comparatively better running in the *RISC* stations than running in the *SUN* stations: RISC architectures have pipelining features which favors the implementation of vector inner products required by PCG methods.

CONCLUSIONS

In order to make meaningful comparisons between direct and iterative methods the same general testing conditions were used with the two competing approaches. A new ordering scheme which brings to the iterative method based on incomplete factorizations the same type of benefits that Tinney-2 scheme brings to the direct methods was suggested and tested. Although in the PCG methods studied in the paper structure is invariant (no fill-ins are considered), it is shown that network parameters may affect convergence. Eigenvalue clustering (and clumping) produced by the new ordering scheme are fully discussed as well.

The paper presents results obtained in comparative studies designed to evaluate the performances of direct versus iterative solvers of $Ax = b$ systems found in power flow calculations and contingency analysis (linear contingency screening with decoupled $P\theta$ and QV half-iterations). The general conclusion is that direct methods are still faster, though iterative methods are easier to implement [19]. Parallelism and vectorization, which normally bring more benefits to the iterative methods than they do to direct methods, have not been investigated.

A new type of pre-conditioner naturally suited to the contingency screening problem was tested: base-case LDU factors, obtained with conventional direct methods, were used to pre-condition $Ax = b$ in contingency screening (single/multiple circuit removals which do not cause disconnections). By a wide margin this conditioner presented the best performance among all pre-conditioners tested (K_0 and incomplete Cholesky). As a matter of fact, this was the only method to show performances comparable with the ones obtained with direct methods; moreover, the proposed approach is competitive even for small networks such as IEEE 57-bus and 118-bus systems.

Although not critical to the final evaluation of the methods, a difference in relative performances was observed when the direct and iterative approaches were tested in *SUN SPARCstation 2* and *IBM-RISC/6000* stations: the iterative method showed to be comparatively better running in the *RISC* stations than running in the *SUN* stations (*RISC* architectures have some pipelining features which favors the implementation of vector inner products required by PCG methods).

Summarizing, the paper presents both theoretical and practical studies comparing direct and iterative methods for solving power flow and contingency analysis problems. Two important contributions are also made for improving the performance of the iterative methods in the proposed applications: (1) a new ordering scheme (*minimum spanning tree* algorithm) for pre-conditioned conjugate gradient algorithms based on incomplete factorizations; and (2) a new pre-conditioning scheme based on complete factorizations and specially suited to the contingency analysis problem.

ACKNOWLEDGMENTS

This research has been supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo), CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), and CNPq (Conselho Nacional de Pesquisas Científicas e Tecnológicas).

REFERENCES

[1] H. Mori, H. Tanaka and J. Kanno, "A Pre-conditioned Fast Decoupled Power Flow Method for Contingency Screening", *Proc. of PICA'95*, May 1995.

[2] I. C. Decker, D. M. Falcão and E. Kaszkurewicz, "Conjugate Gradient Methods for Power System Dynamic Simulation on Parallel Computers", IEEE-PES Summer Meeting, July 1995.

[3] A. Gupta, V. Kumar and A. Sameh, "Performance and Scalability of Preconditioned Conjugate Gradient Methods on Parallel Computers", *IEEE Trans. on Parallel and Distributed Systems*, Vol. 6, No. 5, May 1995.

[4] B. Stott, O. Alsaç and A. Monticelli, "Security Analysis and Optimization", Invited Paper. *Proc. of the IEEE*, Vol. 75, No. 12, pp. 1623-1644, Dec. 1987.

[5] M. K. Enns, J. J. Quada and B. Sacckett, "Fast Linear Contingency Analysis". *IEEE Trans. on PAS*, Vol. 101, No. 4, pp. 783-791, April 1982.

[6] O. Alsaç, B. Stott and W. F. Tinney, "Sparsity-Oriented Compensation Methods for Modified Network Solutions", *IEEE Trans. on PAS*, Vol. 102, pp. 1050-1060, May 1983.

[7] W. F. Tinney, V. Brandwajn and S. M. Chan, "Sparse Vector Methods", *IEEE Trans. on PAS*, Vol. 104, No. 2, pp. 295-301, Feb. 1983.

[8] S. M. Chan and V. Brandwajn, "Partial Matrix Refactorization", *IEEE Trans. on Power Systems*, Vol. PWRS-1, No. 1, pp. 193-200, Feb. 1986.

[9] I. S. Duff and G. A. Meurant, "The Effect of Ordering on Preconditioned Conjugate Gradients", *BIT*, **29**:635-657, 1989.

[10] G. H. Golub and C. F. Van Loan, "Matrix Computations", The Johns Hopkins University Press, USA, 1983.

[11] H. Dağ and F. Alvarado, "The Effect of Ordering on the Preconditioned Conjugate Gradient Method for Power Systems Applications", *Proc. of the North American Power Symposium*, Manhattan, Kansas, pp. 202-209, Sept. 1994.

[12] A. Jennings, "Influence of the Eigenvalue Spectrum on the Convergence Rate of the Conjugate Gradient Method", *J. of the Institute of Mathematics and Applications*, **20**:61-72, 1977.

[13] H. Javidi, S. McFee and F.D. Galiana, "Investigation of Eigenvalue Clustering by Modified Incomplete Cholesky Decomposition in Power Network Matrices", *Proc. of the Power System Computation Conference*, Aug. 1993.

[14] J. A. Meijerink and H. A. van der Vorst, "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems", *J. of Computational Physics*, **44**:134-155, 1981.

[15] D. S. Kershaw, "The Incomplete Cholesky-Conjugate Gradient Method for Iterative Solution of Systems of Linear Equations", *J. of Computational Physics*, **26**:43-65, 1978.

[16] A. George and J. W. Liu, "Computer Solution of Large Sparse Positive Definite Systems", Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[17] F. D. Galiana, H. Javidi and S. McFee, "On the Application of a Preconditioned Conjugate Gradient Algorithm to Power Network Analysis", *Proc. of PICA'93*, pp. 404-410, May 1993.

[18] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. van der Vorst, "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", 2nd Edition, SIAM, Philadelphia, PA, 1994.

[19] A. Semlyen, "Fundamental Concepts of a Krylov Subspace Power Flow Methodology", IEEE-PES Summer Meeting, July 1995.

BIOGRAPHIES

Antonio Cesar Baleeiro Alves received his B.Sc. degree from UFG in 1983, and his M.Sc. degree from UFU in 1991, all Electrical Engineering. He is currently working towards his Ph.D degree at Unicamp.

Eduardo Nobuhiro Asada is an undergraduate student of Electrical Engineering at Unicamp.

Alcir J. Monticelli (Senior Member, IEEE) received his Ph.D. degree from Unicamp in 1975 where he is currently a Professor of Electrical Engineering.

PRODUÇÃO

Artigos Aprovados para Publicação ou Publicados

1. A.B. Alves, E.N. Asada and A. Monticelli, *Critical Evaluation of Direct and Iterative Methods for Solving Ax = b Systems in Power Flow Calculations and Contingency Analysis.* To appear in IEEE Transactions on Power Systems. To be presented in IEEE Summer Meeting, Berlin, Germany, July 24, 1997.

2. R. A. Gallego, A.B. Alves, A. Monticelli and R. Romero, *Parallel Simulated Annealing Applied to Long Term Transmission Network Expansion Planning.* IEEE Transactions on Power Systems, Vol. 12, No.1, pp. 181-188, February 1997.

3. Fujio Sato, Alcir J. Monticelli, Ariovaldo V. Garcia y Antonio C. Baleeiro Alves, *Análisis de Cortocircuito Probabilístico por el Método de Monte Carlo Utilizando Ambientes Paralelo y Distribuído.* Revista Internacional INFORMACION TECNOLOGICA (ISSN:0716-8756), Vol. 8 No. 5, Sept-Oct.,1997.

4. F. Sato, A.C.B. Alves, A.J. Monticelli and A.V. Garcia, *A Comparative Study of Parallel and Distributed Probabilistic Short-Circuit Analysis.* Proceedings of the 7th International Symposium Short-Circuit Currents in Power Systems (SCC), Vol. 1, pp. 1.31.1-1.31.8, Poland, Warsaw, 10-12 September 1996.

5. A.C. Baleeiro Alves and A. Monticelli, *Parallel and Distributed Solutions for Contingency Analysis in Energy Management Systems.* Proceedings of the 38TH Midwest Symposium on Circuits and Systems, pp. 449-452, Brazil, Rio de Janeiro, August 13-16, 1995.

6. A. Monticelli, F. Sato and A.C. Baleeiro Alves, *Parallel and Distributed Computing Applied to the Analysis of Large Scale Electric Power Network Problems.* ANAIS DO SIMPÓSIO NIPO-BRASILEIRO DE CIÊNCIA E TECNOLOGIA-100 ANOS DE AMIZADE BRASIL-JAPÃO, pp. 213-226. Brazil, Campos do Jordão, August 14-16, 1995.

7. A. Monticelli, A. Garcia, F. Sato e A.C. Baleeiro Alves, *Análise de Curto-Circuito Probabilístico pelo Método de Monte Carlo Utilizando Ambientes Paralelo e Distribuído.* ANAIS DO XI CONGRESO CHILENO DE INGENIERIA ELECTRICA, pp. A-053-058 Chile, Punta Arenas, 13 a 17 de Novembro, 1995.

Artigos Submetidos

1. F. Sato, A.V. Garcia, A. Monticelli and A.C. Baleeiro Alves, *Distributed Short-circuit Analysis in Heterogeneous Computer Networks.* International Journal on Electrical Power & Energy Systems. Europa. Submitted in Aug. 1996.

2. A.C. Baleeiro Alves and A. Monticelli, *Static Security Analysis Using Pipeline Decomposition.* IEE Proceedings. Transmission and Distribution. Submitted in Aug. 1996.