APLICAÇÃO DE PROCESSAMENTO PARALELO AO PROBLEMA DE PLANEJAMENTO DA OPERAÇÃO DE SISTEMAS HIDROTÉRMICOS BASEADO EM CLUSTER DE COMPUTADORES

Roberto José Pinto

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientadores: Carmen Lucia Tancredo Borges
Maria Elvira Piñeiro Maceira

Rio de Janeiro
Setembro de 2011

APLICAÇÃO DE PROCESSAMENTO PARALELO AO PROBLEMA DE PLANEJAMENTO DA OPERAÇÃO DE SISTEMAS HIDROTÉRMICOS BASEADO EM CLUSTER DE COMPUTADORES

Roberto José Pinto

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

_____
Prof[a]. Carmen Lucia Tancredo Borges, D.Sc.

_____
Dra. Maria Elvira Piñeiro Maceira, D.Sc.

_____
Prof. Djalma Mosqueira Falcão, Ph.D.

_____
Prof. Álvaro Luiz Gayoso de Azeredo Coutinho, D.Sc.

_____
Prof. Eduardo Nobuhiro Asada, D.Sc.

_____
Prof. Rubén Augusto Romero Lázaro, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2011

Pinto, Roberto José

Aplicação de Processamento Paralelo ao Problema de Planejamento da Operação de Sistemas Hidrotérmicos Baseado em *Cluster* de Computadores / Roberto José Pinto. – Rio de Janeiro: UFRJ/COPPE, 2011.

XXI, 290 p.: il.; 29,7 cm.

Orientadores: Carmen Lucia Tancredo Borges

Maria Elvira Piñeiro Maceira

Tese (doutorado) – UFRJ / COPPE / Programa de Engenharia Elétrica, 2011.

Referências Bibliográficas: p. 219-231.

1. Processamento Paralelo. 2. Computação de Alto Desempenho. 3. Planejamento da Operação Energética. 4. Sistemas Hidrotérmicos. 5. *Cluster* de Computadores. I. Borges, Carmen Lucia Tancredo *et al*. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família Carmem Lúcia, Rafael e Helena.

À minha mãe Beoclésia e à minha sogra Valdete.

A vocês devo tudo.

Agradecimentos

- Às minhas orientadoras, pela paciência, incentivo e valiosos ensinamentos ao longo deste trabalho.

- Ao CEPEL, pelo apoio financeiro e pela infraestrutura computacional.

- Ao NACAD, pela infraestrutura computacional, e em especial ao amigo Albino dos Anjos Aveleda pela constante e sempre prestativa ajuda ao longo deste trabalho.

- À Carmen Lucia Tancredo Borges, Maria Elvira Piñeiro Maceira, Djalma Mosqueira Falcão e Albert Cordeiro Geber de Melo por acreditarem e incetivarem o início deste trabalho.

- Aos amigos de departamento, Vitor Silva Duarte, Débora Dias Jardim Penna, André Luiz Diniz, Luiz Guilherme Marzano, Fábio Rodrigo Siqueira Batista, Michel Pompeu Tcheou, Tiago Norbiato, Ana Lúcia Gouveia de Sabóia, Ana Carolina, Luciano Xavier, Valk Castellani e Carlos Henrique Medeiros de Sabóia pela inestimável colaboração, incentivo e amizade.

- Aos demais amigos do CEPEL, Flávio Rodrigo, Sérgio Porto, Carlos Frederico, Juan Ignácio e Ricardo Diniz pela ajuda e amizade sempre presente.

- A todos aqueles que, direta ou indiretamente, me ajudaram a concluir este trabalho, estejam onde estiverem.

- A todos os meus familiares pelo incentivo e carinho, em especial ao meu pai Wilson (em memória), à minha mãe Beoclésia, à minha sogra Valdete, à minha tia Britznéa (em memória) e ao meu tio Roberto, que me levou a fazer engenharia.

- Um agradecimento especial à minha esposa Carmem Lúcia da Silva, pelo amor, carinho, incentivo, paciência e tolerância, me permitindo encontrar paz e tranqüilidade, condições fundamentais para a execução deste trabalho.

- Outro agradecimento especial à minha filha Helena da Silva Pinto e ao meu enteado Rafael da Silva Rodrigues Vieira pelo amor e carinho. Helena, papai te pede desculpas pelos momentos em que não pôde conviver contigo como deveria.

- A Deus, pela oportunidade de estar aqui e pela ajuda que sempre tive ao longo desta vida.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

APLICAÇÃO DE PROCESSAMENTO PARALELO AO PROBLEMA DE PLANEJAMENTO DA OPERAÇÃO DE SISTEMAS HIDROTÉRMICOS BASEADO EM CLUSTER DE COMPUTADORES

Roberto José Pinto

Setembro/2011

Orientadores: Carmen Lucia Tancredo Borges
Maria Elvira Piñeiro Maceira

Programa: Engenharia Elétrica

Este trabalho tem por objetivo o desenvolvimento de uma estratégia de paralelização aplicada ao problema de planejamento da operação de sistemas hidrotérmicos. Esse problema foi resolvido por programação dinâmica dual estocástica, com a determinação de um plano de operação para cada usina, minimizando o valor esperado do custo total da operação do sistema ao longo do horizonte de planejamento. A cada estágio e para cada estado do sistema (nível de armazenamento e afluências nos meses anteriores), o problema de operação hidrotérmica é modelado como um problema de programação linear e as variáveis duais associadas à sua solução são utilizadas para a construção dos cortes de Benders. O plano de operação é representado pela função de custo futuro (FCF), que é aproximada através de uma função linear por partes, construída iterativamente por estes cortes. No processo de construção da FCF é aplicada uma estratégia de paralelização, pois, em cada estágio e estado do sistema, são resolvidos tantos problemas de despacho de operação quantos forem os cenários de afluência para o período e estes problemas são independentes entre si.

A estratégia de paralelização proposta utiliza as funções da biblioteca MPI para a comunicação entre os processadores, além de possuir um balanceamento dinâmico, de forma a minimizar o tempo ocioso dos processadores. Além disto, diversos procedimentos foram executados de forma a tornar a eficiência final a melhor possível.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

PARALLEL PROCESSING APPLIED TO THE PROBLEM OF HYDROTHERMAL SYSTEMS OPERATION PLANNING BASED ON CLUSTER COMPUTERS

Roberto José Pinto

September/2011

Advisors: Carmen Lucia Tancredo Borges
Maria Elvira Piñeiro Maceira

Department: Electrical Engineering

This work presents the development of a parallelization strategy applied to the problem of hydrothermal systems operation planning. This problem is solved by stochastic dual dynamic programming. A plan of operation is determined for each stage of the planning period and given an initial state, the generation for each plant and the main objective of minimizing the expected total cost of operation over the planning horizon. For each state, hydrothermal operation problem is modeled as a linear programming problem and the dual variables associated with the solution are used to construct the Benders cuts. The plan of operation is represented by the future cost function, which is approximated by a piecewise linear function, constructed iteratively by the Benders cuts. In the process of building the future cost function, a parallelization strategy is applied for each stage and in each state of the system (storage levels and scenarios of inflows for the previous months), can be solved in a independent way.

The proposed parallelization strategy uses the library functions from MPI library for communication between processors, as well as a dynamic balance in the distribution of problems in order to minimize the idle time of processors. In addition, several procedures were performed to improve performance.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas

ANEEL → Agência Nacional de Energia Elétrica.

API → *Application Programming Interface.*

CA → Corrente Alternada.

CC → Corrente Contínua;

CCOI → Comitê Coordenador para Operação Interligada.

CEPEL → Centro de Pesquisas de Energia Elétrica.

CLP → COIN *Linear Programming*. Biblioteca para solução de PLs desenvolvida pela COIN.

COIN → *Computational Infrastructure for Operations Research.*

CPU → *Central Process Unit.* Unidade central de processamento.

Dell → Empresa do ramo de sistemas de computadores.

EM64T → ou x86-64, conjunto de instruções utilizadas nos processadores de 64 bits.

EPE → Empresa de Pesquisa Energética.

FCF → Função de Custo Futuro.

FCP → Função de Custo Presente.

FFT → *Fast Fourier Transform.*

FSB → *Front Side Bus*. Frequência do barramento de memória.

GCOI → Grupo Coordenador para Operação Interligada.

GNU → Projeto de desenvolvimento de software livre (*GNU General Public License)* da *Free Software Foundation.*

GPU → *Graphics Processing Unit.*

HP → *Hewlett-Packard*. Empresa do ramo de sistemas de computadores.

HPF → *High Performance Fortran.*

IBM → *International Business Machines.* Empresa do ramo de sistemas de computadores.

INTEL → *Integrated Electronics Corporation*. Fabricante de semicondutores.

LAM → *Local Area Multicomputer*. Implementação para Utilização de MPI.

MME → Ministério das Minas e Energia.

MPI → *Message Passing Interface*. Padrão para trocas de mensagens em ambientes de processamento distribuído.

MPICH → Implementação para Utilização de MPI.

NASA → *National Aeronautics and Space Administration.*

NVIDIA → Fabricante de semicondutores.

ONS → Operador Nacional do Sistema Elétrico Brasileiro.

OpenMP → *Open Multi-Processing*

OpenPBS → *Open Portable Batch System.* Padrão de sistema de fila.

OSL → *Optimization Subroutine Library.* Biblioteca para solução de PLs desenvolvida pela IBM.

PDDD → Programação Dinâmica Dual Determinística.

PDDE → Programação Dinâmica Dual Estocástica.

PDE → Programação Dinâmica Estocástica.

PL → Problema de programação linear.

PLs → Problemas de programação linear.

PMO → Programação Mensal da Operação.

POSIX → *Portable Operating System Interface for Unix.*

PVM → *Parallel Virtual Machine*.

RAID → *Redundant Array of Independent Disks*. Modo de conectar mais de 1 disco rígido.

RAM → *Random Access Memory*.

SAS → *Serial Attached* SCSI. Padrão para comunicação/conexão de discos rígidos.

SATA → *Serial Advanced Technology Attachment*. Padrão para comunicação/conexão de discos rígidos

SCSI → *Small Computer System Interface*. Padrão para comunicação/conexão de discos rígidos.

SGI → *Silicon Graphics International*. Empresa do ramo de sistemas de computadores.

SMP → *Symmetric Multiprocessors*. Sistemas com multiprocessamento simétrico

Capítulo 1.
Introdução

1.1 Considerações Gerais

Uma grande modificação no método científico utilizado na modelagem dos fenômenos naturais foi causada pelo surgimento do computador. No método clássico, conforme pode ser visto na Figura 1 [1], a observação de fenômenos da natureza fazia com que o cientista formulasse uma teoria que era confrontada com a experimentação física do fenômeno. A confrontação dos resultados propostos na teoria com os experimentais poderia comprová-la, totalmente ou em parte, ou mostrar que a mesma estava errada, fazendo com que a mesma fosse modificada ou melhorada.

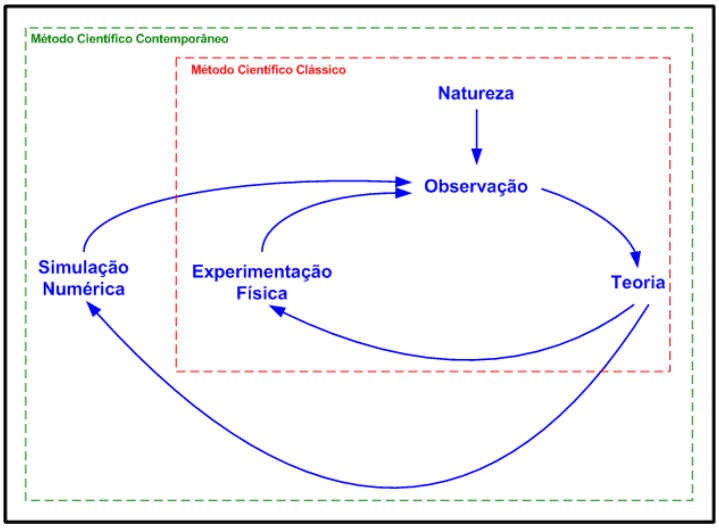

Figura 1 – Diferenças entre os Métodos Científicos Clássico e Contemporâneo

O método clássico possui um grande problema, que é a dificuldade de se realizar determinados experimentos físicos para se obter resultados, ou por serem muito caros,

ou por serem muito perigosos para o homem, ou por conta do fenômeno ser muito difícil de ocorrer na natureza, ou até mesmo por razões éticas. Exemplos deste tipo de dificuldade são encontrados na astronomia, genética, desenvolvimento de armas, experimentos médicos etc.

Com o advento do computador, com a sua constante melhoria de desempenho, houve o desenvolvimento de modelos matemáticos para representar os fenômenos físicos e com o passar do tempo, estes modelos foram ficando cada vez mais precisos e mais rápidos, tornando possível em várias situações a substituição da experimentação física pela simulação do modelo matemático num computador. O cientista moderno pode comparar os resultados da simulação numérica do seu modelo matemático, gerado pela teoria proposta, com os da natureza. As diferenças encontradas servem para que a teoria ou o modelo matemático sejam revistos ou melhorados.

Diante deste novo cenário, pode-se perceber a grande demanda por processamento cada vez mais rápido, uma vez que a diminuição do tempo para se obter resultados acelera o desenvolvimento de teorias e a precisão dos modelos matemáticos. Muitos problemas científicos necessitam de modelos matemáticos extremamente complexos, que por sua vez necessitam de computadores com grande capacidade de processamento para que a pesquisa possa ser viável. Estes problemas complexos são chamados de grandes desafios pela ciência e estão inseridos nas seguintes categorias [1]:

1. Química quântica, mecânica estatística e relatividade física;
2. Cosmologia e astrofísica;
3. Dinâmica de fluidos e turbulência;
4. Desenvolvimento de materiais e supercondutividade;
5. Biologia, farmacologia, projeto de seqüenciamento de genomas, engenharia genética, processo químico de enovelamento de proteínas, cinética enzimática e modelagem celular;
6. Medicina e modelagem de órgãos e ossos humanos;
7. Previsão de tempo e modelagem climática.

Além destas, existem inúmeras outras aplicações que demandam processamento de alto desempenho, tais como: processamento gráfico; manipulação de matrizes;

sistemas de equações lineares; transformadas de Fourier; processos de busca e classificação; otimização do planejamento e operação de sistemas hidrotérmicos; prospecção de petróleo etc.

Para todos estes problemas, o aumento da capacidade de processamento convencional disponibilizado pela indústria pode não ser suficiente para atender suas necessidades. Uma alternativa viável é a computação paralela, uma vez que ela consegue acelerar em muitas vezes a capacidade de processamento. Pode-se afirmar que a computação paralela tem a propriedade de antecipar o futuro, uma vez que a capacidade de processamento necessária para solucionar determinados problemas, que só estaria disponível alguns anos à frente com o aumento da capacidade de processamento dos atuais processadores, pode ser obtida com a ligação de vários computadores para executar o processamento de tarefas em paralelo.

Por conta desta crescente demanda por processamento, do barateamento dos processadores e aumento da velocidade dos equipamentos de rede, do desenvolvimento de ferramentas padronizadas para desenvolvimentos de aplicações paralelas, criou-se em 1994 [1] uma forma de associação de computadores que se tornou muito comum atualmente, que é o chamado *cluster*[1]. Originalmente artesanal, seu sucesso foi quase que instantâneo, já que conseguia, com um conjunto de computadores pessoais comuns e custo muito mais baixo, um poder de processamento parecido com o de computadores proprietários muito mais caros. A própria indústria se rendeu e hoje em dia todos os grandes fabricantes, IBM, HP, Dell, SGI etc., oferecem opções deste tipo de computador. Para se ter uma idéia deste sucesso, na última edição dos 500 maiores computadores do mundo em novembro de 2010 [2], 414 (82,8%) eram *clusters*.

Além da demanda por computadores mais poderosos, e também por causa dela, a sociedade, não só a científica, exige cada vez mais demanda por energia. Atualmente, praticamente tudo que utilizamos necessita de energia elétrica. A dependência da sociedade é tão grande que, sem energia elétrica, nem água teríamos, uma vez que as grandes bombas hidráulicas necessitam dela para funcionar.

---

[1] *cluster* consiste num agrupamento de computadores, interligados através de rede, que trabalham conjuntamente, trocando informações entre si. As tarefas a serem executadas são repartidas entre os computadores, que trabalham como se fossem uma única máquina.

No Brasil, a principal forma de geração de energia elétrica é através de usinas hidráulicas, de forma a aproveitar a grande quantidade de energia fluvial existente. Segundo o boletim de monitoramento mensal, publicado pelo Ministério de Minas e Energia [3], em abril de 2011 dos 114.229MW de capacidade instalada, 81.152MW (71%) eram provenientes de usinas hidráulicas, 30.141MW (26,4%) de origem térmica convencional (gás, biomassa, petróleo e carvão mineral), 2.007MW (1,8%) de origem térmica nuclear, 929MW (0,8%) de origem eólica e 0,09MW proveniente de energia solar.

De forma a aproveitar de maneira racional toda essa energia, este sistema hidrotérmico precisa de ferramentas para planejar e operar de maneira ótima sem desperdiçar recursos, uma vez que sistemas, com uma porcentagem substancial de geração hidroelétrica, podem utilizar a energia armazenada "grátis" nos reservatórios do sistema para atender a demanda, substituindo assim a geração dispendiosa das unidades térmicas. Entretanto, o volume de água afluente aos reservatórios é desconhecido, pois depende basicamente das afluências que irão ocorrer no futuro. Além disso, a disponibilidade de energia hidroelétrica é limitada pela capacidade de armazenamento dos reservatórios. Este fato introduz uma relação entre uma decisão de operação em um determinado período e as conseqüências futuras desta decisão. Por exemplo, se for decidido utilizar energia hidroelétrica para atender o mercado hoje e no futuro ocorrer uma seca, poderá ser necessário utilizar geração térmica de custo elevado ou, até, ser interrompido o fornecimento de energia. Por outro lado, se for utilizado de forma mais intensiva a geração térmica, conservando elevados os níveis dos reservatórios, e ocorrerem vazões altas no futuro, poderá haver vertimento no sistema, o que representa um desperdício de energia e, conseqüentemente, um aumento desnecessário no custo de operação. Estas situações estão apresentadas na Figura 2.

O objetivo básico do planejamento da operação energética de um sistema hidrotérmico é determinar, para cada etapa do período de planejamento, as metas de geração para cada usina que atendam a demanda e minimizem o valor esperado do custo de operação ao longo do período. No caso do Sistema Elétrico Brasileiro, devido as suas características, tais como, comportamento estocástico das vazões afluentes às usinas hidroelétricas, existência de grandes reservatórios, geração afastada dos centros

consumidores e sistema interligado, esse problema é de grande porte, sendo a coordenação hidrotérmica uma tarefa que exige grande esforço computacional [4].

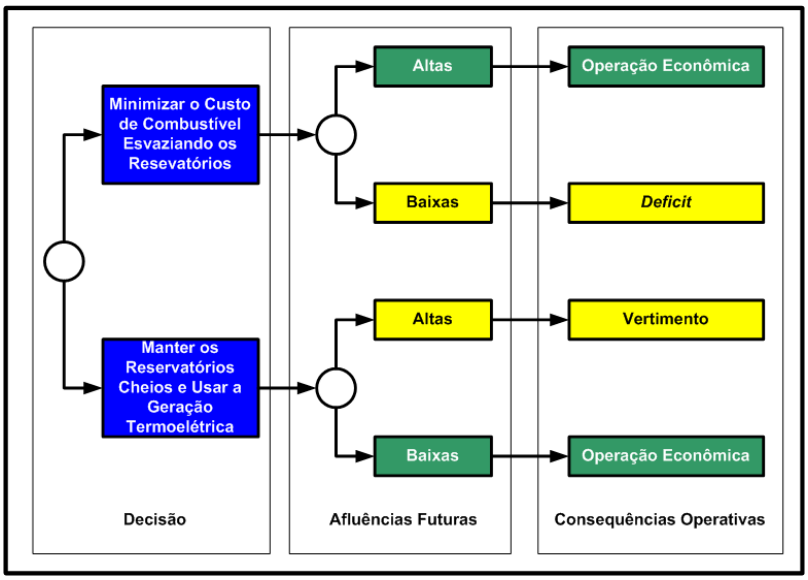

Figura 2 – Processo de Decisão para Sistemas Hidrotérmicos

1.2 Objetivo deste Trabalho

A determinação das metas de geração das usinas que minimizam o custo total de operação no problema de planejamento do Sistema Elétrico Brasileiro é um processo que demanda um intenso processamento computacional, uma vez que esse sistema é de grande porte, possuindo uma grande quantidade de usinas hidráulicas e térmicas interligadas entre si. Por conta disto, existe uma grande demanda para que os tempos totais gastos na obtenção da solução ótima sejam reduzidos. Este problema pode ser resolvido por programação dinâmica dual estocástica (PDDE) [5], [6] e [7], que utiliza um processo iterativo constituído por tarefas que são independentes entre sí, indicando que a aplicação de técnicas de programação paralela farão com que o tempo de solução do problema diminua significativamente. O objetivo deste trabalho é propor e aplicar uma metodologia, que permita a utilização, de processamento paralelo no problema de planejamento da operação do Sistema Elétrico Brasileiro, reduzindo significativamente o tempo total gasto na obtenção da sua solução. É dada ênfase à análise da eficiência da estratégia de paralelização proposta de modo a aumentá-la o máximo possível.

1.3 Breve Histórico do Planejamento da Operação do Sistema Hidrotérmico Brasileiro

O início da geração de energia elétrica no Brasil aconteceu no final do século 19 com a entrada em operação da primeira usina hidrelétrica, localizada no rio Ribeirão do Inferno, afluente do rio Jequitinhonha, na cidade de Diamantina, com a construção da primeira termelétrica em Campos, ambas em 1883, e a instalação da primeira usina hidrelétrica, usina Marmelos-zero em Juiz de Fora, destinada ao abastecimento público em 1889 [8].

Este sistema foi crescendo ao longo dos anos, porém era pouco interligado, sendo marcado pela proliferação de sistemas isolados e concentração de usinas próximas às cidades de São Paulo e Rio de Janeiro em meados do século passado [9]. A interligação de sistemas começou na década de 1920, com o objetivo de transferir energia em épocas de crise, não possuindo nenhuma operação coordenada.

A primeira grande área a ser interligada foi o interior paulista, por intermédio da Companhia Paulista de Força e Luz. Outro exemplo de interligação de sistema foi feito pela Light ao conectar as usinas de Cubatão, em São Paulo, e Fontes, no Rio de Janeiro. Na década de 1950 a interligação dos sistemas avançou bastante graças aos governos estaduais, principalmente de Minas Gerais e São Paulo, com seus planos de eletrificação. Na década de 1960 começou a era das grandes usinas com a entrada em operação de Três Marias (1962) e Furnas (1963). Em 1964, a freqüência de 60Hz foi adotada como padrão em todo o país. A interligação entre os sistemas da região sudeste foi uma necessidade, tanto para receber as energias geradas pelas grandes usinas, construídas longe dos centros urbanos, quanto para melhor utilizar os recursos hídricos disponíveis, evitando colapsos no fornecimento de energia às indústrias instaladas principalmente nos estados de São Paulo e Rio de Janeiro.

Nas outras regiões, as interligações dos sistemas elétricos ocorreram de forma diversa: na região sul, a primeira interligação foi feita em 1967 entre Paraná e Santa Catarina; na região nordeste, a Chesf já interligava todas as capitais da região no final da década de 1960; na região norte, por conta de características locais, os sistemas permaneceram isolados e supridos por sistemas térmicos.

Em 1969 foi criado o CCOI, abrangendo as empresas geradoras e distribuidoras da região Sudeste, do qual a Eletrobrás foi a coordenadora técnica. Em 1971 foi criado o CCOI da Região Sul.

Em 1973 foi criado o GCOI. Seus objetivos eram obter uma melhor utilização dos recursos hidráulicos e térmicos, realizar estudos elétricos e energéticos abrangendo o horizonte de cinco anos à frente e coordenar a operação do sistema em tempo real [10].

Entre 1974 e 1978, a estratégia de operação do sistema foi calculada através de um critério determinístico, baseado no registro histórico das afluências médias mensais. O objetivo era utilizar a energia térmica disponível no sistema para garantir o atendimento do mercado na ocorrência do pior cenário histórico de afluências. Este critério foi chamado de Método da Curva Limite [10], pois o mesmo leva à determinação de uma curva limite inferior de armazenamento do sistema, indicando o nível mínimo do armazenamento necessário para atender o mercado até o final do horizonte de planejamento, sem ocorrência de *deficit* e com qualquer cenário de afluências históricas.

A partir de 1979, o GCOI passou a utilizar as estratégias de operação calculadas por um modelo computacional de otimização baseado na técnica de programação dinâmica estocástica (PDE), desenvolvido em conjunto pelo CEPEL e Eletrobrás entre 1975 e 1977 [11]. De 1979 até 1984, a utilização deste modelo computacional permitiu uma redução de 28% no custo de operação dos sistemas interligados brasileiros, representando uma economia de US$260 milhões [12].

Por conta de restrições da PDE, onde a dimensão do problema cresce exponencialmente com o número de variáveis de estado, em 1986, o modelo computacional foi substituído por outro baseado na programação dinâmica dual estocástica (PDDE) [13].

Conforme exposto por Soares [14], a complexidade do problema do planejamento da operação de sistemas hidrotérmicos exige uma estratégia de decomposição que permita explorar suas características físicas através da modelagem matemática. Para prazos curtos, por exemplo, as incertezas com relação às afluências são pequenas, podendo-se dizer que os problemas são quase que determinísticos, uma vez que a previsão de demanda do mercado também pode ser estimada com uma boa

precisão. Para prazos muito longos, a incerteza em relação às afluências cresce até atingir a saturação, representando a independência estocástica entre as vazões futura e presente. Para prazos entre estes dois limites, a incerteza é crescente, porém menor que a de longo prazo. Por conta destas características, é recomendável tratar o problema do planejamento da operação em curto, médio e longo prazos.

Atualmente, o Operador Nacional do Sistema (ONS), órgão criado em 1998 para substituir o GCOI durante a reforma do Setor Elétrico Brasileiro, utiliza uma cadeia de modelos computacionais, desenvolvidos pelo CEPEL, para tratar das diversas etapas do planejamento da operação [4].

1.4 Revisão Bibliográfica

1.4.1 Aplicações de Computação Paralela em Sistemas Elétricos

Neste item estão descritos vários trabalhos de aplicação de computação paralela em sistemas de potência. Por estes trabalhos percebe-se que a utilização de técnicas de processamento paralelo é bastante vasta, uma vez que existem trabalhos em confiabilidade, análise de contingências, fluxo de potência ótimo, fluxo de potência, recomposição de sistema, transitórios eletromagnéticos, curto-circuito, estabilidade de pequenos sinais, estimação de estado, harmônicos em sistemas de potência etc. Nestes casos, a aplicação de técnicas de processamento paralelo pode ser feita através da observação de tarefas independente entre si presentes no algoritmo de solução original. Outra forma é a simplificação dos problemas, permitindo o desacoplamento de determinadas tarefas e facilitando o desenvolvimento da aplicação de programação paralela. Outra alternativa é desenvolver um novo método de solução para o problema que permita a utilização de processamento paralelo. Os trabalhos mostrados a seguir, que estão sucintamente descritos, apresentam alguma destas possibilidades de aplicação de técnicas de processamento paralelo.

Em [15], Tylavsky *et al.* apresentaram as principais características de processamento paralelo, além das oportunidades de aplicação destas técnicas de alto desempenho em aplicações de sistemas de potência.

Em [16], Falcão apresenta uma visão geral e os impactos da aplicação de processamento paralelo na área de sistemas de potência. São abordadas soluções para várias aplicações, tais como: análise de estabilidade transitória; transientes eletromagnéticos; estabilidade de pequenos sinais, estimação de estados, fluxo de potência etc.

Em [17], Ferlin abordou conceitos e definições de processamento paralelo que são aplicados à paralelização automática, além de análises e condições para as dependências de dados com o objetivo de aplicar os métodos de paralelização conhecidos como hiperplano, transformação unimodular, alocação de dados sem comunicação e particionamento & rotulação.

Em [18], Feltrin propôs uma metodologia para a decomposição da etapa de solução direta de um sistema esparso de equações lineares, permitindo a aplicação de técnicas de processamento paralelo. Neste trabalho foi utilizado um computador desenvolvido no Centro de Pesquisas e Desenvolvimento da Telebrás, chamado de processador preferencial paralelo (PPP).

Em [19], Vale, Falcão e Kaszkurewicz apresentaram uma metodologia de decomposição para solução de equações de rede de sistemas de potência em ambientes de processamento paralelo.

Em [20], Lau, Tylavsky e Bose apresentaram dois métodos para paralelização da fatoração triangular (LU) da matriz de equações lineares em sistemas de potência.

Em [21], [22], [23] e [24], Borges *et al.* apresentaram metodologias assíncronas de processamento paralelo para avaliação da confiabilidade composta, usando simulação de Monte Carlo não-seqüencial e seqüencial.

Em [25], Teixeira *et al.* aplicaram uma nova abordagem para implementação de técnicas de programação paralela em algoritmos para solução de problemas de confiabilidade, avaliação de segurança de sistemas hidrotérmicos e fluxo de potência ótimo com restrições de segurança.

Em [26], Borges, Falcão e Taranto descreveram implementações de técnicas de processamento paralelo em problemas de análise da confiabilidade composta da geração e transmissão de sistemas de potência, de análise de segurança dinâmica e de utilização de algoritmos genéticos para ajuste de estabilizadores. Foram apresentados resultados obtidos num *cluster* de computadores pessoais.

Em [27], Moreira Junior utilizou diretivas de OpenMP para paralelizar o modelo computacional PLANTAC, que consiste num programa para planejamento da transmissão usando o valor econômico da confiabilidade. Neste modelo é feita uma análise probabilística durante a investigação da necessidade de adição de reforços na transmissão de um sistema de potência. Dependendo do porte do sistema a ser investigado, o processo combinatório dos possíveis reforços pode se tornar bastante demorado, implicando na necessidade da utilização de técnicas de processamento paralelo. Neste trabalho foram executados vários casos e as eficiências sempre se situaram em valores superiores a 75%, com a utilização de processadores com quatro núcleos de processamento.

Em [28], Alves utilizou técnicas de paralelização para viabilizar o uso de fluxo de potência ótimo com restrições de segurança na operação em tempo real. Este trabalho também foi publicado por Borges e Alves em [29]

Em [30], Alves utilizou técnicas de processamento paralelo para o problema de análise de contingência e cálculo de fluxo de potência ótimo com restrições de segurança. Na implementação paralela foi utilizada a biblioteca de rotinas PVM.

Em [31], Méndez apresentou um algoritmo paralelo para a solução do fluxo de potência ótimo com restrições de segurança. Os resultados foram obtidos através da utilização do algoritmo proposto no computador PPP, desenvolvido no Centro de Pesquisas e Desenvolvimento da Telebrás, e também num computador nCUBE2.

Em [32], Decker, Falcão e Kaszkurewicz apresentaram resultados de testes de uma implementação, com processamento paralelo, de uma metodologia para análise de estabilidade transitória. Neste método, as equações de rede foram resolvidas através da combinação da fatorização LU com o método do gradiente conjugado.

Em [33], Decker, Falcão e Kaszkurewicz apresentaram três métodos, baseados em paralelizações espaciais ou temporais, para simulações de problemas de análise de estabilidade transitória de sistemas de potência em ambientes de processamento paralelo. Com relação ao tema análise de estabilidade transitória, diversos outros trabalhos foram publicados, tais como: Em [34], La Scala *et al.* apresentaram um novo método para simulação de problemas de estabilidade transitória. Este método procura aproveitar ao máximo as oportunidades de paralelização que o problema oferece. Em [35], La Scala *et al.* apresentaram um algoritmo paralelo, chamado de *pipelined-in-time*

parallel algoritm, para resolver problemas de estabilidade transitória, baseado no método de Newton para resolver as equações que descrevem o sistema em cada passo de integração. Além disso, o algoritmo de Gauss-Seidel é adotado para relaxar a solução, desacoplando os intervalos de tempo, permitindo a análise independente dos passos de integração nos diversos processadores. Em [36], Lee *et al.* desenvolveram um sistema computacional rápido e eficiente para análise de estabilidade transitória em uma configuração hipercubo multiprocessada. Para realizar a comunicação entre os processadores de forma eficiente, foi inserida uma pseudo-árvore binária. Esta falsa árvore é uma estrutura binária onde um nó no hipercubo pode corresponder a mais de um nó na árvore binária correspondente. O objetivo desta pseudo-árvore é executar mais eficientemente as funções que envolvam a comunicação de dados entre os processadores. Foram executados dois casos, um com a matriz de admitâncias cheia e outro com a matriz de admitâncias esparsa. O caso com a matriz de admitâncias cheia é encarado com um limite, uma vez que a mesma é comumente bastante esparsa. Os resultados apresentaram uma eficiência acima de 80%, para a matriz cheia, e acima de 70%, para o caso com a matriz esparsa, para as execuções com 8 ou menos processadores, caindo significativamente nos casos com mais quantidades de processadores. Em [37], Chai *et al.* testaram algoritmos paralelos do tipo Newton para simulações de estabilidade transitória em computadores com arquiteturas de memória distribuída e compartilhada. Em [38], Castro investigou a viabilidade da utilização de ambientes de memória distribuída para avaliar o nível de segurança dinâmica de sistemas elétricos de potência em tempo real. O método de análise utilizou PVM para gerenciar e coordenar as trocas de mensagens que fazem com que os processos se comuniquem entre si. Aplicação de processamento paralelo em estabilidade transitória também é o assunto principal dos trabalhos [39], [40], [41], [42] e [43]

Em [44], Vieira Junior propôs um método paralelo para solução do cálculo de fluxo de potência trifásico em redes assimétricas de distribuição de energia elétrica. Este método baseia-se no método de solução de Z_{bus} Gauss, e no desacoplamento das fases abc através de injeção de correntes. Com o desacoplamento, as equações de cada fase podem ser resolvidas de forma independente, permitindo o uso de computação paralela para acelerar a solução do problema. Foram feitos testes num computador paralelo nCUBE2, com 64 nós, e com a utilização do PVM (*Parallel Virtual Machine*).

Conforme sugerido por Alves [45], a paralelização pode muito bem ser aplicada ao problema de recomposição do sistema elétrico, mais especificamente na busca de corredores viáveis, principalmente se a função de avaliação envolver a análise de transitórios eletromecânicos.

Roméro [46] utilizou a decomposição lagrangeana na solução do problema de planejamento da expansão de sistemas de transmissão de energia elétrica, considerando cenários de incertezas. Neste trabalho utilizou-se a relaxação lagrangeana, que possibilita a transformação de problemas grandes e difíceis em uma seqüência de problemas mais fáceis que podem ser resolvidos em paralelo.

Em [47], Oliveira desenvolveu diversas metaheurísticas combinatórias para a solução do problema do planejamento da expansão da transmissão dos sistemas de energia elétrica. Este problema foi analisado de forma estática e a longo prazo, incluindo uma versão paralela da metodologia *simulated annealing*, além de diversas versões paralelas de algoritmos genéticos. Para as implementações paralelas foi utilizada a biblioteca de rotinas PVM. Foram obtidos resultados com o sistema Garver, sistema sul brasileiro, sistema norte-nordeste e o sistema Colombiano.

Em [48], Palin aplicou técnicas de decomposição de domínio e processamento paralelo na solução de grandes sistemas de equações algébricas lineares resultantes da modelagem de fenômenos eletromagnéticos pelo método de elementos finitos. No ambiente de processamento foi utilizada a biblioteca MPI.

Em [49], Trevizan implementou uma solução paralela para simular o comportamento de fenômenos eletromagnéticos utilizando o método das diferenças finitas no domínio do tempo. Nesta implementação foi utilizada a biblioteca de comunicação LAM/MPI.

Em [50], Falcão, Kaszkurewicz e Almeida utilizaram técnicas de processamento paralelo para solução de problemas de transitórios eletromagnéticos em sistemas de potência.

Em [51], Falcão, Wu e Murphy exploraram a possibilidade de implementação de processamento paralelo em problemas de estimação de estado, muito utilizado em programas avançados de sistemas de gerenciamento de energia.

A aplicação de técnicas de processamento paralelo para solução de problemas de estabilidade de pequenos sinais em sistemas de potência está apresentada em [52], por Campagnolo *et al.*, e em [53], por Campagnolo, Martins e Falcão.

Em [54], Sato paralelizou um programa de análise de curto-circuito probabilístico utilizando o método de Monte Carlo. A implementação paralela utilizou rotinas da biblioteca PVM. Este trabalho também foi publicado por Sato, Garcia e Monticelli em [55].

Em [56], Tão tratou do problema de minimização de perdas em sistemas de distribuição de energia elétrica. A solução proposta envolveu técnicas de otimização de fluxos não lineares em redes, com a utilização do método do gradiente reduzido, e procedimentos da área de inteligência artificial. A implementação paralela utilizou rotinas da biblioteca PVM.

Em [57], Barán, Kaszkurewicz e Falcão destacaram o uso da combinação de diferentes métodos para resolver sistemas complexos de equações algébricas para aplicação em cálculos de fluxo de carga de sistemas de potência. Estes times de algoritmos são utilizados em sistemas paralelos de memória distribuída e com comunicação assíncrona.

Em [58], Almeida utilizou o modelo de tempos assíncronos (*Times-A*), em ambiente de processamento paralelo, para a determinação de soluções iniciais para o processo de planejamento da expansão da transmissão. A implementação paralela utilizou rotinas da biblioteca PVM.

Em [59], Mariños, Pereira e Carneiro Jr. apresentaram uma nova abordagem para os estudos de harmônicos em sistemas de potência. Baseado no fato de que o acoplamento entre os harmônicos de diferentes ordens é desprezível, os autores adotaram técnicas de processamento paralelo para resolver este problema.

Em [60], Aveleda desenvolveu e utilizou um *cluster* de baixo custo, baseado em computadores comuns, para resolver problemas de processamento de sinais relacionados com engenharia *offshore*. Neste trabalho, mostrou-se um ganho de desempenho da ordem de 100, combinando um novo algoritmo de solução para análise de vibrações induzidas com processamento paralelo.

Em [61], Wu-Zhi *et al.* apresentam um algoritmo paralelo para cálculo de autovalores em sistemas de potência no domínio da freqüência. Este cálculo é de

extrema importância para análise da estabilidade de pequenos sinais para identificar fenômenos de oscilações de baixa freqüência entre áreas. Este algoritmo aproveita a capacidade de multiprocessamento dos processadores atuais através do uso de *threads*. Foram apresentados resultados com base num subsistema da China com 36648 variáveis de estado.

Em [62], McGinn e Shaw apresentam um algoritmo para executar a fatoração de Matrizes através do método de eliminação de Gauss em ambientes de processamento paralelo. São apresentados resultados com versões utilizando MPI, para ambientes com memória distribuída, e utilizando OpenMP, para ambientes com memória compartilhada.

Em [63], Shi *et al.* apresentam uma solução paralela para o método iterativo de Jacobi para resolver sistemas de equações lineares. Inicialmente a implementação paralela utilizou apenas instruções MPI e posteriormente uma segunda versão, hibrida, utilizando MPI e OpenMP foi desenvolvida para aproveitar a arquitetura de memória compartilhada existente nos processadores recentes com vários núcleos de processamento, diminuindo a comunicação entre os nós. Os resultados mostraram que a versão hibrida foi significantemente mais eficiente para um caso cuja matriz de coeficientes do sistema Ax=b era 5000x5000, num sistema com 8 nós e processadores com dois núcleos de processamento.

Em [64], Faria Junior apresenta uma nova metaheurística, denominada GRAPR (*Greedy Randomized Adaptive Search Procedure and Path Relinking*), para a solução de problemas do tipo otimização combinatória. Esta metaheurística foi aplicada ao problema do planejamento da expansão de sistemas de transmissão de energia elétrica, tendo sido desenvolvida uma versão com processamento paralelo, utilizando instruções de MPI, de forma a reduzir os tempos gerados pela análise combinatorial do processo.

Em [65], Marcato *et al.* aplicaram algoritmos genéticos para resolver o problema do planejamento da expansão da geração de longo prazo. A solução para este problema envolve a utilização de programação linear inteira e possui duas grandes características: região de solução não convexa, permitindo várias soluções; natureza combinatorial, que tende a apresentar uma tendência à explosão combinatorial, por conta do grande aumento das alternativas possíveis de solução. Este tipo de problema de otimização pode ser resolvido com a aplicação de técnicas de computação evolutiva, onde se

destaca o algoritmo genético. Os autores informaram que a metodologia proposta foi eficiente para um caso teste e a redução de tempo computacional foi quase linear quando a estratégia de paralelização foi aplicada num caso do sistema Brasileiro, indicando que a eficiência do método foi bastante elevada.

Em [66], Matar e Iravani apresentaram uma modelagem massivamente paralela para representação de máquinas CA para simulações em tempo real de transientes eletromagnéticos. O paralelismo foi explorado no algoritmo de solução do modelo de máquina, uma vez que o passo de integração da simulação é muito menor que as constantes de tempo dos sistemas elétrico e mecânico, o acoplamento entre estes sistemas pode ser removido através da inclusão de um passo de tempo entre a solução dos dois conjuntos de equações. Desta forma, os dois sistemas de equações podem ser resolvidos de forma independente. Os autores também utilizaram uma aproximação válida ao desacoplar as correntes de estator dos eixos "d" e "q".

Em [67], Li, Li e Li, apresentaram uma solução paralela para o cálculo de fluxo de potência utilizando o método da inversão da matriz Ybus e o algoritmo de Newton-Raphson para resolver o sistema de equações. Os autores particionaram a rede e usaram um algoritmo para corrigir as equações de cada subrede, resolvendo o sistema de forma paralela. Para um dos sistemas, utilizado como exemplo, composto por 9768 barras, 6515 linhas de transmissão CA, 7264 transformadores, 2 linhas de transmissão CC, 1048 geradores e 2904 cargas, a solução com 4 subredes apresentou o melhor resultado em termos de fator de aceleração (4,2) e a solução com 2 subredes apresentou a melhor eficiência (90%).

Em [68], Li *et al.* apresentaram um algoritmo paralelo híbrido para calcular FFT (*Fast Fourier Transform*). É mostrado que a solução híbrida, utilizando MPI e OpenMP, diminuiu o consumo de memória e melhorou a distribuição de carga entre os processadores por conta da diminuição da comunicação entre os processadores, do que a versão utilizando apenas MPI. Os resultados mostraram que o algoritmo tem boa escalabilidade e alta eficiência.

Em [69], Jia-An, Na e Yi-Lang analisaram métodos paralelos para solução de sistemas de equações com matrizes esparsas de redes elétricas de sistemas de potência. Por conta do crescimento dos processadores com vários núcleos de processamento, os

autores recomendam a adoção de uma programação hibrida, com a adoção do MPI e do OpenMP.

É importante ressaltar o aumento recente de trabalhos recomendando a utilização de uma programação híbrida, com o uso simultâneo de instruções MPI, para as comunicações entre os nós, e OpenMP, para as tarefas intra-processador, aproveitando o uso da memória compartilhada através do uso de *threads*. Estes trabalhos analisam a utilização deste tipo de programação, tanto em máquinas paralelas [70], [71], [72] e [73], quanto em *clusters* baseados em GPU (*Graphics Processing Unit*) [74] e [75].

1.4.2 Sistemas Hidrotérmicos

Existem vários trabalhos a respeito da aplicação de programação dinâmica estocástica (PDE) para resolver o problema do planejamento da operação de sistemas hidrotérmicos. Nesta linha estão os trabalhos [76], [77], [78], [79] e [80].

Em [76], Mo, Hegge e Wangensteen modelaram, além da dinâmica do sistema, as incertezas na demanda de energia e nos preços da transmissão para problemas de planejamento da expansão. A metodologia proposta foi baseada na PDE e foram apresentados resultados para o sistema norueguês.

Em [77], Dapkus e Bowe formularam uma metodologia para o problema do planejamento da expansão utilizando a PDE. As incertezas da demanda, da entrada de novas tecnologias de combustível de usinas térmicas e possibilidade de perda de serviço de usinas nucleares foram modeladas. Foram apresentados resultados para um sistema hipotético operando num período de seis anos.

Em [78], Quintana e Chikhani propuseram uma metodologia para otimizar a descarga dos reservatórios utilizando a PDE. Foram apresentados resultados para um sistema composto de 17 usinas hidráulicas e 11 usinas térmicas.

Em [79], Sherkat *et al*. utilizaram a PDE para resolver o problema de planejamento da operação de longo prazo de sistemas hidrotérmicos. O algoritmo de solução utilizou a metodologia de PDE com dois estágios a fim de se obter o mínimo custo de operação. Foram apresentados resultados para um sistema composto de 60 usinas térmicas e 35 usinas hidráulicas.

Em [80], Li, Yan e Zhou apresentaram uma metodologia para resolver o problema do planejamento ótimo da operação de longo prazo de sistemas hidráulicos com afluências estocásticas. O subproblema hidráulico foi resolvido com a utilização da PDE com dois estágios, levando em consideração séries temporais para as afluências dos rios. Foi desenvolvido um programa computacional e foram apresentados resultados para os sistemas central e oriental da China, composto de nove reservatórios.

Em [81], Bond propôs uma metodologia determinística para otimização do sistema hidrotérmico interligado sudeste-sul. O objetivo desta metodologia é minimizar o custo operativo plurianual do sistema, representado pelo custo de déficit e do custo dos combustíveis das usinas termoelétricas. As usinas hidroelétricas e termoelétricas são modeladas, assim como o intercâmbio entre os submercados, e as restrições dinâmicas de volume de espera, defluência e volume mínimos. Para a solução do problema é utilizado o método simplex para determinar as estratégias operativas mensais das usinas individualizadas. Este trabalho também foi publicado em [82].

Em [12], Kligerman aplicou o método de programação dinâmica estocástica dual para os subsistemas sudeste e sul, comparando os resultados com outros métodos de planejamento da operação do sistema hidrotérmico brasileiro.

Em [83], Marques, Cigona e Soares expuseram os benefícios da operação coordenada do Sistema Hidroelétrico Brasileiro.

Em [84], Cruz Junior apresenta um modelo equivalente não linear para a etapa de longo prazo do planejamento da operação energética de sistemas hidroelétricos. Na solução é adotada a programação dinâmica estocástica e o sistema hidráulico é agregado através de um modelo equivalente.

Em [85], Zambelli apresentou uma política operativa baseada no conceito de curvas-guia de armazenamento para o planejamento da operação energética de médio prazo. Nesta política operativa, as vazões turbinadas de cada usina hidrelétrica são determinadas levando-se em conta níveis pré-estabelecidos dos reservatórios por curvas-guia de armazenamento.

Em [86], Cigona desenvolveu um programa, orientado a objetos, para resolver o problema de planejamento da operação de sistemas hidrotérmicos de médio e longo prazo, com representação individualizada das usinas.

Em [87], Martinez aborda uma metodologia que combina modelos de otimização determinísticos com modelos de previsão de vazão, numa estrutura de malha aberta. O objetivo foi obter uma solução ótima para o problema de planejamento da operação energética. Este trabalho foi também publicado por Martinez e Soares em [88].

Em [89], Martinez e Soares apresentaram um estudo comparativo entre a programação dinâmica estocástica dual e primal aplicada ao problema de planejamento hidrotérmico. Foram utilizados dados das afluências históricas e a variável estocástica do problema foi modelada por um modelo auto-regressivo de ordem um.

Em [90], Siqueira realizou um estudo comparativo entre a programação dinâmica estocástica e a programação dinâmica dual estocástica, apresentando vantagens e desvantagens destes métodos, considerando o caso particular de sistemas com apenas uma usina hidrelétrica.

Em [91], Matos analisou comparativamente as modelagens de reservatório equivalente por submercados e por cascata no problema de planejamento da operação energética de médio prazo do sistema hidrotérmico brasileiro.

Em [92], Pereira e Pinto descreveram uma metodologia para coordenação do planejamento de curto e médio prazo de sistemas hidrotérmicos. Com esta metodologia, é possível transladar o problema elétrico existente no problema de curto prazo como uma restrição no problema de médio prazo.

Em [93], Souza avaliou o impacto da representação mais detalhada das bacias hidrográficas no problema de planejamento da operação do sistema hidrotérmico brasileiro de médio prazo. Atualmente a modelagem utilizada é a representação das usinas de forma equivalentada por submercado. A representação proposta aumenta o grau de detalhamento do sistema com a representação das usinas equivalentadas por bacias.

Em [94], Lopes analisou o desempenho de diferentes funções objetivo (maximizar a produção energética, minimizar a complementação de energia e maximizar o lucro sobre a energia secundária), assim como regras de deplecionamentos dos reservatórios em paralelo e em série na otimização da operação de sistemas hidráulicos. Os resultados tiveram como base um sistema hipotético de reservatórios.

Em [95], Marques apresenta uma nova política para o planejamento da operação do sistema hidrotérmico brasileiro. Esta política está baseada na representação das

usinas individualizadas, com representação detalhada das suas características de operação, e na representação da estocasticidade das vazões através de um modelo de previsão.

A aplicação de técnicas de computação evolutiva em problemas de otimização é a característica principal dos trabalhos [96], [97], [98] e [99], descritos a seguir.

Como um problema de coordenação hidrotérmica pode ser formulado como um problema de minimização de custos não-linear, o trabalho proposto por Amendola [96] avaliou e comparou a aplicação das seguintes metaheurísticas para a solução deste problema: algoritmos genéticos; enxame de partículas; e recozimento simulado. Foram feitos testes num sistema-teste composto de sete usinas hidrelétricas e seis usinas termelétricas flexíveis e apresentou-se uma comparação do desempenho de cada uma delas.

Em [97], Humpiri aplicou técnicas de computação evolutiva, utilizando algoritmos genéticos, para a solução do problema de planejamento da operação de sistemas hidrotérmicos.

A aplicação de algoritmos genéticos para a solução de otimização da operação de sistemas hidrotérmicos também foi objeto do trabalho de Leite, Carneiro e Carvalho [98]. Os resultados foram apresentados para dois casos, um com 19 usinas com reservatórios e outro com 35 usinas, sendo 19 com reservatórios e 16 a fio d'água. Os autores ainda destacaram a possibilidade de paralelização do método aplicado.

Em [99], Zoumas *et al.* também aplicaram algoritmos genéticos para a solução do planejamento da operação de sistemas hidrotérmicos. O algoritmo genético é aplicado ao subproblema de programação da geração hidráulica. O método proposto foi aplicado num sistema composto de 13 usinas hidráulicas e 28 usinas térmicas.

Em [100], Sifuentes e Vargas apresentaram técnicas para acelerar a solução dos problemas de planejamento de sistemas hidrotérmicos com decomposição de Benders. Esta redução se dá no número de iterações da convergência e no próprio processo de solução.

Em [101], Chang, Fu e Marcus apresentaram um novo algoritmo para solução de problemas de PDE com horizonte finito de tempo. Este algoritmo, que os autores chamaram de "*Simulated Annealing Multiplicative Weights*", é "assintoticamente

eficiente", uma vez que a solução é cada vez mais próxima do valor ótimo conforme o aumento do número de iterações.

Em [102], Unsihuay, Marangon-Lima e Souza propuseram o planejamento da operação de curto-prazo integrada do sistema hidrotérmico e a rede de gás natural, obtendo a otimização dos dois sistemas simultaneamente. O modelo proposto leva em consideração as restrições do sistema hidrotérmico, assim como as restrições de extração, armazenamento e condução do sistema de gás natural. Para otimizar este sistema integrado foram utilizadas decomposição dual, relaxação lagrangeana e programação dinâmica.

1.4.3 Aplicação de Computação Paralela no Planejamento da Operação de Sistemas Hidrotérmicos

Uma aplicação inicial de técnicas de processamento paralelo no problema de planejamento de longo prazo da operação de sistemas hidrotérmicos, utilizando como algoritmo de solução a PDDE, foi feita por Finardi [103] e por Silva e Finardi [104] e [105].

O algoritmo de solução da PDDE consiste num processo iterativo de dois ciclos com a solução de diversos PLs: em um deles, a função que representa o custo de operação dos períodos futuros, decorrente de uma decisão no período atual é montada através de diversos cortes de Benders, que restringem a solução do problema; no outro, estes cortes são utilizados para calcular o custo de operação. No primeiro ciclo, o processo de solução é realizado do final para o início do estudo, sendo chamado de ciclo *backward*. No segundo, o processo de solução é realizado do início para o final do estudo, sendo chamado de ciclo *forward*.

Como na PDDE existem conjuntos de problemas que podem ser resolvidos de forma independente, tanto no ciclo *forward* quanto no ciclo *backward*, os autores distribuíram estes conjuntos por diversos processadores com o objetivo de diminuir o tempo total de solução do problema. No ciclo *forward*, o primeiro a ser executado neste trabalho, os problemas foram distribuídos pelos processadores dividindo-se a quantidade de cenários pela quantidade de processadores disponíveis, ou seja, foi aplicada uma distribuição estática dos problemas pelos processadores. Neste ciclo o

processamento foi feito sem comunicação entre os processadores. No ciclo *backward*, a mesma estratégia foi aplicada, porém, diferentemente do ciclo anterior, existe um ponto de sincronismo provocado pela necessidade de envio dos cortes de Benders, um para cada cenário, de todos os processadores para o processador mestre. De posse de todos os cortes, este processador os distribui para todos os outros, de forma que todos os problemas do período a ser resolvido em seguida tenham as mesmas restrições geradas pelos cortes de Benders do período resolvido imediatamente antes.

O algoritmo proposto pelos autores foi aplicado num problema com uma representação simplificada do Sistema Elétrico Brasileiro, composto de 15 usinas hidráulicas e seis usinas térmicas, com a demanda fixa em 20.400MW durante todo o período de estudo e as funções de produção das usinas hidráulicas foram consideradas constantes. O volume inicial de cada reservatório foi adotado em 50% do volume máximo operativo.

Para a análise de desempenho da estratégia de paralelização, foi utilizada uma configuração de quatro computadores IBM Risc System/6000 Workstations. O caso utilizado possuía 100 cenários, com 15 aberturas de afluências no ciclo *backward*. O período de estudo foi de 24 meses e o número de iterações foi limitado em quatro. Os valores das eficiências totais foram os seguintes: 91,5% (2 processadores), 83,5% (3 processadores) e 78,6% (4 processadores). As eficiências para o ciclo *forward* foram as seguintes: 85,8% (2 processadores), 85,1% (3 processadores) e 84,6% (4 processadores). Para o ciclo *backward*, as eficiências foram as seguintes: 88,2% (2 processadores), 81,3% (3 processadores) e 78,2% (4 processadores).

Ao longo deste trabalho será apresentada uma metodologia para a otimização de uma estratégia de paralelização aplicada à solução de problemas de planejamento da operação de sistemas hidrotérmicos com a utilização de PDDE, permitindo a obtenção de eficiências mais elevadas e de uma menor deterioração das mesmas com o aumento da quantidade de processadores.

1.5 Estrutura do Trabalho

Este trabalho está composto de oito capítulos.

O segundo capítulo diz respeito à formulação matemática do problema de planejamento de sistemas hidrotérmico. São destacados os métodos de soluções utilizando a programação dinâmica estocástica (PDE) e a programação dinâmica dual estocástica (PDDE).

O terceiro capítulo apresenta as características da computação paralela, incluindo arquiteturas de computadores, modelos de programação paralela e os cuidados que devem ser levados em consideração na implementação de programas com processamento paralelo.

O quarto capítulo descreve a estratégia de paralelização inicialmente proposta e implementada para a solução do problema do planejamento da operação em ambiente de processamento paralelo.

O quinto capítulo descreve o processo de otimização da estratégia de paralelização proposta, através do desenvolvimento de versões obtidas com a identificação das causas de queda de desempenho computacional, encontrando soluções que permitam a solução do problema sempre com tempos finais menores, melhorando a eficiência final do processamento paralelo.

O sexto capítulo apresenta os resultados obtidos com a última versão do processo de otimização da estratégia de paralelização proposta. É analisada a aderência dos resultados, bem como o desempenho da metodologia paralela proposta.

O sétimo capítulo apresenta as conclusões deste trabalho e as propostas para sua futura evolução.

Todas as referências estão apresentadas no capítulo oito.

Além destes oito capítulos, também fazem parte deste trabalho três anexos: no anexo A estão apresentados detalhes da implementação da estratégia de paralelização inicial; no anexo B estão apresentados alguns detalhes sobre as implementações para a otimização da estratégia de paralelização; no anexo C estão apresentados detalhamentos dos resultados finais dos casos utilizados neste trabalho.

1.6 Publicações Relacionadas

Em [106] e [107], Borges e Pinto apresentaram uma metodologia para levar em consideração as incertezas de vazões na confiabilidade da geração de pequenas usinas.

Parte da metodologia proposta neste trabalho foi apresentada por Pinto, Duarte e Maceira em [108]. Neste trabalho foram apresentadas as linhas gerais da proposta inicial e também mostrados os resultados obtidos com a sua aplicação no modelo NEWAVE, programa computacional desenvolvido no CEPEL e que será detalhado mais adiante, num cluster com trinta e dois processadores.

Em [109], Pinto, Borges e Maceira apresentaram a metodologia inicial proposta deste trabalho, com os resultados baseado na versão 14 do modelo NEWAVE executado no *cluster* do CEPEL com até 48 processadores.

Além destes trabalhos, a estratégia de paralelização desenvolvida neste trabalho foi aplicada em outro modelo computacional do CEPEL. Em [110] e [111], Pinto, Sabóia *et al.* apresentaram uma estratégia de paralelização similar aplicada ao modelo DECOMP, modelo computacional desenvolvido no CEPEL, utilizado no planejamento de curto-prazo. A implementação de processamento paralelo neste modelo inicialmente possibilitou a execução de casos com um mês determinístico e um mês estocástico [110], e depois a metodologia foi ampliada para a execução de casos com vários meses estocásticos [111], algo inviável, pelo longo tempo de execução demandado, de ser executado pela versão convencional do programa. Os resultados mostraram que os casos de PMO, com um mês determinístico e um mês estocástico, passaram a ser executados em menos de 15 minutos, alguns em torno de 6 minutos, utilizando 32 processadores. Já o tempo de execução de um caso com dois meses estocásticos caiu de quase 20 horas para 36 minutos, com a utilização de 80 processadores, ou 1 hora com a utilização de 32 processadores.

Capítulo 2.
Planejamento da Operação de Sistemas Hidrotérmicos

Neste capítulo estão apresentados os elementos que compõem os sistemas hidrotérmicos, tais como as usinas hidráulicas e térmicas, como estas usinas são modeladas nos problemas de planejamento de operação de sistemas hidrotérmicos, como estes problemas são formulados e a utilização da programação dinâmica estocástica e a programação dinâmica dual estocástica para resolvê-lo.

2.1 O Problema de Planejamento da Operação

O problema de planejamento da operação é tratado de forma diferente para três situações específicas: sistemas puramente térmicos; sistemas hidrotérmicos, com predominância de energia hídrica; e operação de sistemas interligados.

2.1.1 Sistemas Puramente Térmicos

Em sistemas de geração puramente térmicos, ou seja, compostos somente de unidades geradoras térmicas, o custo de cada usina depende basicamente do custo do combustível. Logo, o problema de operação consiste na determinação da combinação de usinas que minimize o custo total de combustível (carvão, óleo, gás, nuclear etc.) necessário para atender a demanda do mercado.

Na versão mais simples, este problema é resolvido colocando-se as usinas em ordem crescente de custo incremental de geração. Porém existem fatores que tornam este problema mais complexo, tais como, limitações de carregamento de linhas de transmissão, custos de partida das unidades geradoras etc.

O problema de operação de um sistema térmico tem as seguintes características básicas:

- É desacoplado no tempo, ou seja, uma decisão de operação hoje não tem efeito no custo futuro de operação;
- As unidades têm um custo direto de operação, ou seja, o custo de operação de uma unidade independe do nível de geração de outras unidades. Além disso, a operação de uma unidade não afeta a capacidade de geração ou a disponibilidade de outra unidade;
- A confiabilidade do fornecimento de energia depende somente da capacidade total de geração disponível e não da estratégia de operação das unidades do sistema.

2.1.2 Sistemas Hidrotérmicos

Ao contrário dos sistemas puramente térmicos, sistemas com geração hidráulica podem utilizar a energia "grátis" através do volume de água armazenada nos reservatórios do sistema, substituindo assim a geração das usinas térmicas, de custo mais elevado. Este tipo de sistema possui duas características principais: (1) o volume de água afluente aos reservatórios é desconhecido, pois depende basicamente das afluências que irão ocorrer no futuro; (2) a quantidade disponível de energia hidroelétrica é limitada pela capacidade de armazenamento dos reservatórios. Estas características fazem com que exista um acoplamento entre a decisão de operação de um determinado período e as conseqüências futuras desta decisão. Por exemplo, se a energia hidroelétrica for maciçamente utilizada hoje e no futuro ocorrer uma seca, a probabilidade da utilização da geração térmica, ou até mesmo da interrupção do fornecimento de energia, cujo custo é ainda maior do que o da geração térmica, será maior. Por outro lado, se a geração térmica for utilizada de forma mais intensiva, mantendo os níveis dos reservatórios elevados, e ocorrerem vazões altas no futuro, talvez seja necessário verter a água armazenada, o que significa um desperdício de energia. Estas situações estão apresentadas na Figura 2, mostrada no capítulo anterior.

O problema de planejamento da operação de sistemas hidrotérmicos possui as seguintes características:

- É acoplado no tempo, ou seja, é necessário avaliar as conseqüências futuras de uma decisão no período corrente. A solução ótima é um equilíbrio entre o benefício do uso atual da água e o benefício futuro de seu armazenamento, medido pelo valor esperado da economia de uso dos combustíveis das unidades térmicas;
- É um problema estocástico, já que existem incertezas com relação às vazões afluentes e à demanda de energia;
- É acoplado no espaço, ou seja, existe uma dependência na operação de usinas hidroelétricas de uma mesma cascata, uma vez que a quantidade de água liberada em uma usina afeta a operação das outras usinas situadas a jusante;
- O valor da energia hidráulica gerada não pode ser obtido de forma direta, sendo medido através da economia resultante nos custos de geração térmica ou nos cortes de fornecimentos evitados (*deficit*);
- Existe um antagonismo com relação aos objetivos de economia de operação e da confiabilidade de atendimento. Isto ocorre porque a máxima utilização da energia hidráulica é a política mais econômica, pois minimiza o uso dos combustíveis das usinas térmicas. Porém, esta política é a menos confiável, porque aumenta o risco de falta de energia armazenada no futuro, aumentando o risco de *deficit*. Já a máxima confiabilidade de fornecimento é obtida mantendo-se elevados os níveis dos reservatórios. Entretanto, isto implica na maior utilização de geração térmica e, por conseguinte, num maior custo de operação. O equilíbrio entre estes dois parâmetros é obtido através do valor do custo do *deficit*, que representa o impacto econômico da interrupção do fornecimento. A determinação deste custo é um problema muito complexo, porém fundamental para o cálculo da política de operação mais adequada para o sistema. Se este custo for muito baixo, haverá uma maior utilização da geração hidráulica, diminuindo os níveis dos reservatórios, provocando maiores riscos de racionamento no futuro. Se o custo do *deficit* for muito alto, haverá uma maior utilização da geração térmica, aumentando o custo de operação do sistema.

2.1.3 Operação de Sistemas Interligados

A existência de interligações com outros sistemas permite uma redução dos custos de operação e um aumento da confiabilidade do fornecimento, uma vez que passa

a ser possível a utilização da energia armazenada de outro sistema para suprir uma deficiência local.

Porém, para a minimização do custo da operação de um sistema interligado, o mesmo deve operar de forma integrada. O cálculo isolado da política ótima por cada empresa, não resulta na operação mais econômica possível. Dessa forma, para obter os ganhos operativos máximos de um sistema hidrotérmico interligado, é necessário operar esse sistema de maneira integrada, otimizando conjuntamente a operação de todos os subsistemas, com o objetivo de minimizar o custo de operação total.

2.2 Modelagem das Usinas Hidráulicas

Uma usina hidrelétrica pode ser representada através das características hidráulicas, conforme mostrada na Figura 3 [112]. Nesta representação pode-se observar os componentes do balanço hidráulico, que são: a vazão afluente que chega ao reservatório; as perdas por evaporação e infiltração; a vazão turbinada, responsável pela geração de energia elétrica; e vazão vertida, que não gera energia elétrica; e a vazão defluente, que é aquela que sai da usina.

A operação destas usinas deve respeitar o princípio de conservação de massa. Desta forma, o volume de água que entra numa usina deve ser igual à soma do volume de saída com as perdas por evaporação e infiltração mais o volume que foi acumulado no reservatório. Pelo mesmo princípio, o volume de saída da usina deve ser igual à soma do volume de água turbinado com o volume vertido [87] e [113].

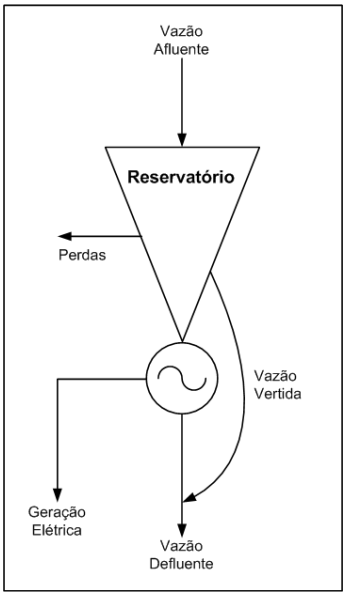

Figura 3 – Modelagem de Uma Usina Hidráulica com Reservatório de Acumulação

Ao se dividir os volumes de água por uma unidade de tempo, tem-se o princípio de conservação instantânea da água, no qual as variáveis passam a ser expressas em termos de vazões, conforme está mostrado na seguinte expressão.

$$Q_d(i,t) = Q_t(i,t) + Q_v(i,t) \tag{1}$$

Onde:

- $Q_d(i,t)$ → Vazão defluente da usina "*i*" em um dado instante de tempo "*t*";
- $Q_t(i,t)$ → Vazão turbinada pela usina "*i*" em um dado instante de tempo "*t*";
- $Q_v(i,t)$ → Vazão vertida pela usina "*i*" em um dado instante de tempo "*t*".

A vazão turbinada é aquela que passa pelo conjunto turbina/gerador produzindo energia elétrica. Esta vazão possui um limite superior definido pelo engolimento máximo, que é a vazão máxima que pode passar pelos condutos forçados até chegar à turbina, e também possui um valor mínimo definido por critérios de construção do conjunto turbina/gerador e por outros usos que o rio possa ter, tais como: navegação, irrigação etc. Por conta destas limitações, a vazão turbinada da usina "*i*" fica limitada conforme está apresentada na expressão a seguir.

$$Q_{min}(i) \geq Q_t(i,t) \geq Q_{max}(i) \tag{2}$$

Com relação ao volume de água que chega ao reservatório [87], este pode ser proveniente do próprio recurso hídrico (rio, lagos etc.) ou da área de drenagem da bacia hidráulica. A vazão afluente pode ter duas parcelas [87]: natural e incremental. A vazão afluente natural é o volume total de água que passa por uma seção transversal do rio, considerando-se todas as descargas das usinas hidráulicas a montante, tanto laterais como do próprio rio. A vazão afluente incremental é o volume de água que chega ao rio devido à área de drenagem a montante da seção transversal considerada.

A vazão defluente de uma usina será uma das parcelas da vazão afluente da usina imediatamente a jusante. Logo, a vazão afluente de uma usina pode ser obtida através de seguinte expressão:

$$Q_a(i,t) = \sum_{m \in M(i)} [Q_t(m,t) + Q_v(m,t)] + Q_l(i,t) \tag{3}$$

Onde:

- $Q_a(i,t) \rightarrow$ Vazão afluente da usina "i" em um dado instante de tempo "t";
- $M(i) \rightarrow$ Conjunto de usinas situadas a montante da usina "i";
- $Q_t(m,t) \rightarrow$ Vazão turbinada pela usina "m" em um dado instante de tempo "t";
- $Q_v(m,t) \rightarrow$ Vazão vertida pela usina "m" em um dado instante de tempo "t";
- $Q_l(i,t) \rightarrow$ Vazão lateral da usina "i" em um dado instante de tempo "t".

A altura de queda bruta de uma usina pode ser definida como:

$$h_b(i,t) = \Psi(i, vol(i,t), Q_d(i,t)) = FCM(i, vol(i,t)) - FCJ(i, Q_d(i,t)) \tag{4}$$

Onde:

- $h_b(i,t) \rightarrow$ Altura de queda bruta da usina "i" em um dado instante de tempo "t";
- $vol(i,t) \rightarrow$ Volume do reservatório da usina "i" em um dado instante de tempo "t";
- $Q_d(i,t) \rightarrow$ Vazão defluente da usina "i" em um dado instante de tempo "t";
- $\Psi(i, vol(i,t), Q_d(i,t)) \rightarrow$ Função altura de queda bruta da usina "i";
- $FCM(i, vol(i,t)) \rightarrow$ Função cota montante. Esta função relaciona o volume do reservatório com a altura da lâmina d'água do mesmo em um dado instante de tempo "t";

- $FCJ(i,Q_d(i,t))$ → Função cota jusante. Esta função relaciona a vazão defluente da usina com a altura da lâmina d'água na saída da usina em um dado instante de tempo "*t*".

As funções FCM() e FCJ() são funções definidas através de polinômios para representar o efeito não linear da variação da cota (lâmina d'água) do reservatório com o volume do mesmo (FCM) e da variação da cota do canal de fuga com a vazão defluente (FCJ).

Para se calcular a altura líquida é necessária a determinação das perdas hidráulicas da passagem da água pelas tubulações de pressão e forçada e pelos órgãos adutores da turbina. Estas perdas serão levadas em consideração através de uma redução da altura de queda bruta do reservatório da usina.

A altura líquida da usina pode ser determinada pela seguinte expressão.

$$h_l(i,t) = h_b(i,t) - h_p(i,t) \tag{5}$$

Onde:

- $h_l(i,t)$ → Altura da queda líquida da usina "*i*" em um dado instante de tempo "*t*";
- $h_b(i,t)$ → Altura da queda bruta da usina "*i*" em um dado instante de tempo "*t*";
- $h_p(i,t)$ → Altura da queda equivalente, representando as perdas hidráulicas da usina "*i*" em um dado instante de tempo "*t*";

Normalmente numa usina hidráulica existem mais de um conjunto gerador/turbina (*NoCjMaq*), que podem ser semelhantes ou não, sendo que conjuntos semelhantes podem ser agrupados e conter várias unidades (*NoMaq*). Para cada unidade gerador/turbina de um conjunto é definida uma potência efetiva (*PotEf*), que é a máxima potência ativa que pode ser gerada em regime permanente de operação. A menor queda d'água líquida que permite com que o conjunto gerador/turbina gere a potência efetiva é chamada de queda efetiva, ou altura efetiva (h_{ef}). A vazão turbinada que, submetida à queda efetiva, produz a potência efetiva é chamada de vazão efetiva ou engolimento efetivo (q_{ef}).

A partir destes valores, a potência efetiva e a vazão efetiva da usina podem ser calculadas através das expressões a seguir.

$$POT\_EF(i) = \sum_{j=1}^{NoCjMaq(i)} PotEf(i,j) NoMaq(i,j) \tag{6}$$

Onde:

- $POT\_EF(.)$ → Potência efetiva da usina;
- $NoCjMaq(.)$ → Número de conjunto de máquinas da usina;
- $PotEf(.,.)$ → Potência efetiva de cada conjunto de máquinas existente na usina;
- $NoMaq(.,.)$ → Número de máquinas de cada conjunto existente na usina.

$$Q\_EF(i) = \sum_{j=1}^{NoCjMaq(i)} Q_{ef}(i,j) NoMaq(i,j) \tag{7}$$

Onde:

- $Q\_EF(.)$ → Vazão efetiva da usina;
- $NoCjMaq(.)$ → Número de conjunto de máquinas da usina;
- $Q_{ef}(.,.)$ → Potência efetiva de cada conjunto de máquinas existente na usina;
- $NoMaq(.,.)$ → Número de máquinas de cada conjunto existente na usina.

A potência disponível de um aproveitamento hidráulico depende do peso específico da água, da vazão do rio, da altura líquida do empreendimento e dos rendimentos da turbina e do gerador elétrico, conforme pode ser visto na expressão a seguir.

$$P(i,t) = \gamma\, Q_t(i,t)\, h_l(i,t)\, \eta_T(i, Q_t(i,t))\, \eta_G(i) \tag{8}$$

Onde:

- γ→ peso específico da água. É igual a $1000\frac{Kg}{m^3} \times 9{,}81\frac{N}{Kg} = 9810\frac{N}{m^3}$;
- $Q_t(i,t)$ → Vazão turbinada pela usina "*i*" em um dado instante de tempo "*t*";
- $h_l(i,t)$ → Altura da queda líquida da usina "*i*" em um dado instante de tempo "*t*";
- η_T → Rendimento da turbina da usina "*i*";
- η_G → Rendimento do gerador da usina "*i*".

Com relação ao rendimento da turbina, é importante ressaltar que o mesmo depende do tipo de turbina, além de vários parâmetros construtivos. Além disso, o rendimento da turbina também é variável em relação à vazão turbinada. Porém, a turbina é projetada para operar numa faixa de vazões em que o rendimento seja

aproximadamente constante e, para usinas com reservatório de acumulação, pode-se considerar que a vazão esteja regularizada, o que implicaria numa operação com valores de vazões muito próximos. Por estas razões, pode-se considerar que o rendimento da turbina do aproveitamento é praticamente constante ao longo do regime de operação normal.

Em relação ao rendimento do gerador, tal qual ocorre na turbina, ele é dependente dos parâmetros construtivos. Como ele é uma máquina síncrona, pode-se considerar insignificante a variação de velocidade do eixo, mantendo-se o rendimento constante.

Com a simplificação do rendimento da turbina e considerando um instante de tempo qualquer, a equação (8) pode ser expressa da seguinte forma.

$$P(i) = \gamma\, Q_t(i)\, h_l(i)\, \eta_T(i)\, \eta_G(i) \tag{9}$$

Esta equação pode ser reescrita da seguinte forma.

$$P(i) = \rho(i)\, Q_t(i) \tag{10}$$

Onde $\rho(i)$ é chamado de produtibilidade da usina "*i*" e é definido pela seguinte expressão.

$$\rho(i) = \gamma\, h_l(i)\, \eta_T(i)\, \eta_G(i) \tag{11}$$

2.3 Modelagem das Usinas Térmicas

Em estudos de planejamento, a representação das usinas térmicas é feito através de características construtivas, tais como, potência máxima, combustível utilizado etc., ou de restrições operativas, tais como, nível mínimo de geração, taxa de tomada de carga etc.

Uma característica importante das usinas térmicas que é levada em consideração em estudos de planejamento é a variação do custo de combustível em função da potência gerada pela usina. Na Figura 4 [114], está mostrada uma curva típica do custo de geração em função da potência gerada de uma usina térmica. Esta curva é obtida através dos cálculos de projeto, sendo que o valor mínimo de geração está geralmente

relacionado com a estabilidade do ciclo de combustão utilizado pela usina, sendo considerado uma restrição de projeto da geração térmica. Com relação à potência máxima da usina, as turbinas utilizadas não possuem elevada capacidade de sobrecarga, fazendo com que a geração máxima não ultrapasse muito, em geral apenas 5%, o valor nominal definido pelo fabricante [114].

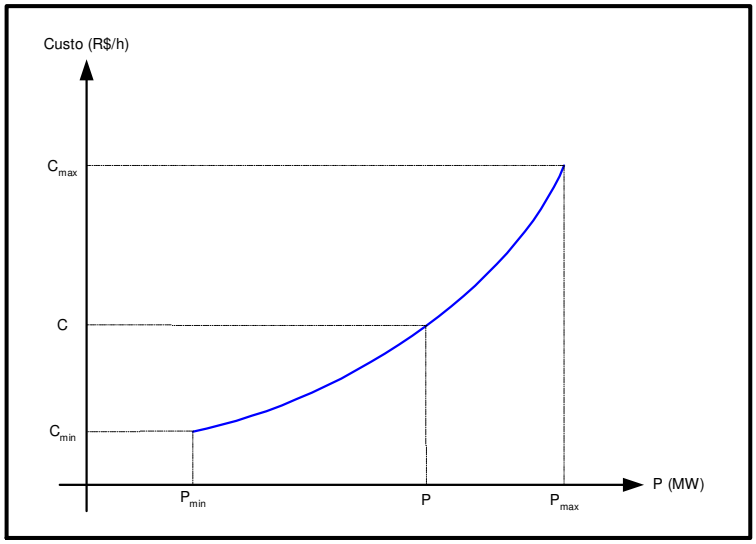

Figura 4 – Custo de Produção Típico de Uma Usina Térmica

Estas restrições operativas das usinas térmicas, devem ser levadas em conta na formulação do problema de planejamento da operação de sistemas hidrotérmicos. Com relação ao custo de produção de energia da usina, um parâmetro que entra na formulação do problema, na parte do custo da usina térmica, é o custo incremental, que consiste no custo necessário para aumentar de uma unidade a potência de saída. A obtenção do custo incremental de uma usina é obtido através da diferenciação da função que fornece o custo de produção em relação à potência de saída. Esta função geralmente é bem aproximanada por uma função de segundo grau, o que implica que a sua derivada é uma função de primeiro grau. O coeficiente ângular desta reta é o chamado custo incremental e ele possui a característica de ser modelado de forma constante no problema do planejamento da operação.

Para grandes turbinas a vapor, existem sistemas de admissão com uma ou várias válvulas que são abertas em sequência para obter um aumento de potência da unidade térmica. Quando uma destas válvulas é aberta, ocorre um repentino aumento do consumo de combustível, fazendo com que o custo incremental da usina não seja mais

constante, podendo até mesmo ser uma função descontínua em relação à potência de saída. Esta modelagem se torna muito mais complexa e normalmente não é feita porque a função do custo incremental deixa de ser convexa, o que é conflitante com as técnicas normalmente utilizadas para solucionar o problema de otimização, que são baseadas na convexidade das funções modeladas.

O uso da modelagem utilizando o custo incremental da usina térmica é utilizada para todos os tipos de usinas térmicas, sejam elas convencionais ou nucleares.

2.4 Etapas do Planejamento da Operação de Sistemas

O planejamento ótimo da operação de um sistema hidrotérmico consiste em se obter as metas de geração de cada usina, hidráulica e térmica, com o mínimo custo ao longo de um determinado período. É um problema de otimização estocástica, não-linear, de grande porte, com acoplamento temporal e espacial e com sistemas interconectados. É necessário obter valores para o sistema desde vários anos até o despacho horário, logo, é impossível resolver este problema na sua totalidade de uma forma única. Por conta destas características, pode-se adotar a decomposição deste problema em etapas [12], onde pode-se modelar mais detalhadamente as características determinantes de cada uma delas, conforme pode ser visto na Figura 5.

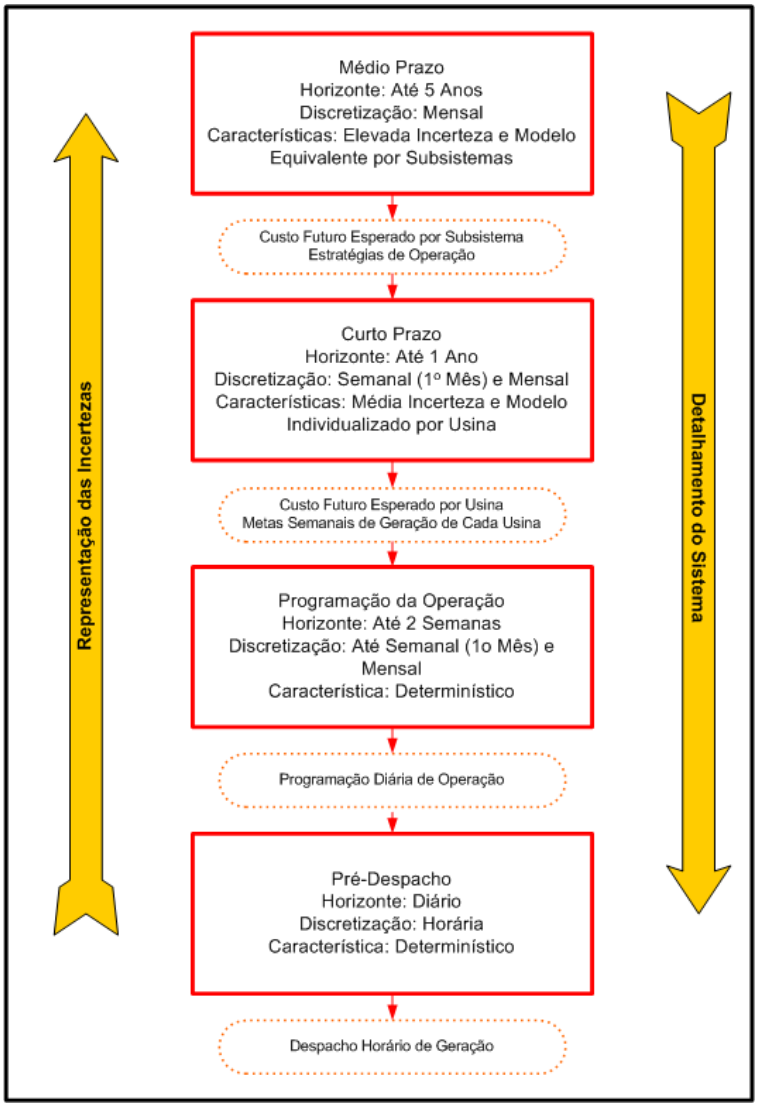

Figura 5 – Etapas do Planejamento da Operação

Neste trabalho, o foco estará centrado na etapa de médio prazo, onde se obtém as estratégias de operação mensais para um horizonte de planejamento de até cinco anos.

2.5 Sistema Equivalente

Como o sistema hidrotérmico brasileiro é de grande porte, pode ser adotada a simplificação de equivalentar os reservatórios [115] em um ou em vários equivalentes, de forma a diminuir o problema, e o tempo de solução, do planejamento da operação de médio e longo prazo. Esta simplicação é válida para os estudos de médio e longo prazo,

onde a precisão da representação individualizada das usinas perde importância em relação às incertezas das vazões futuras [12]. Porém esta não é a única forma de abordar este problema, podendo-se adotar a representação individual das usinas [116] ou uma representação híbrida [117].

A simplificação do sistema implica na operação paralela dos reservatórios, isto é, os armazenamentos e deplecionamentos são feitos paralelamente em volume, ou seja, todos os reservatórios das usinas de um mesmo aproveitamento operam no mesmo sentido ao mesmo tempo. Esta característica é válida para um conjunto de reservatórios com pouca diversidade hidrológica. Outra característica da simplificação é a utilização de variáveis energéticas em vez de variáveis hidráulicas equivalentes ao conjunto de reservatórios agregados. Na Figura 6 estão mostrados os principais componentes de um sistema equivalente.

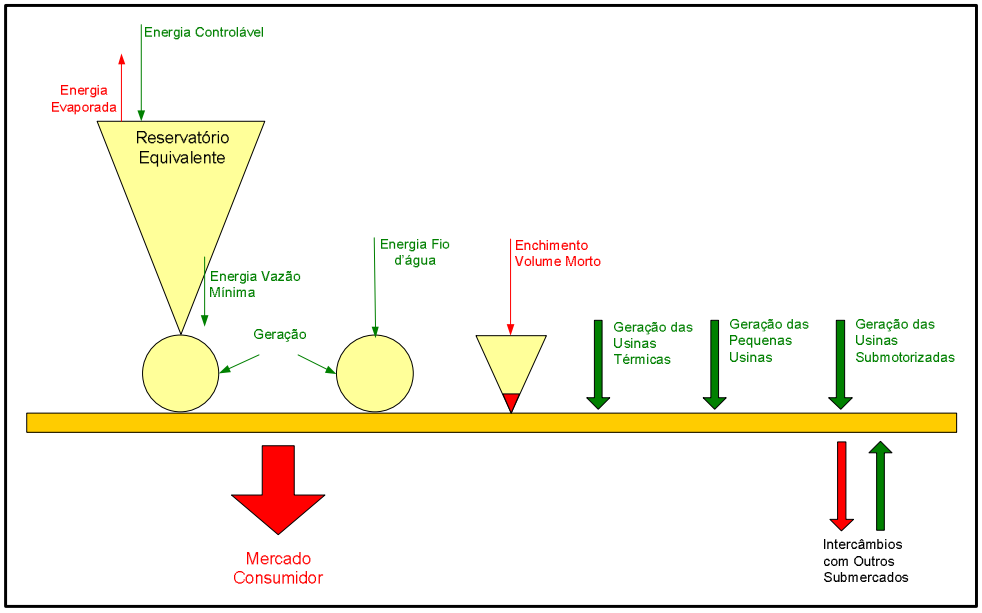

Figura 6 – Componentes do Sistema Equivalente

A seguir serão apresentados os diversos componentes do sistema equivalente [12], [113] e [118].

2.5.1 Energia Armazenável Máxima

A energia armazenável máxima é a capacidade total de armazenamento do conjunto de reservatórios de um subsistema. Ela pode ser estimada pela energia

produzida pelo esvaziamento completo dos reservatórios do subsistema, adotando-se a hipótese de operação em paralelo, isto é, todos os reservatórios das usinas de um mesmo aproveitamento operam no mesmo sentido ao mesmo tempo, tanto no caso de armazenamento quanto no de deplecionamento. Sendo assim, a energia armazenada entre dois estados de armazenamento é definida como sendo a energia gerada ao se deplecionar paralelamente os reservatórios entre os estados inicial e final, sem considerar novas afluências.

A energia armazenável máxima do reservatório equivalente é calculada através da soma dos produtos dos volumes úteis de cada reservatório pelas respectivas produtibilidades de todas as usinas a jusante do mesmo, uma vez que a água que foi utilizada para gerar energia em uma usina viajará ao longo do rio e será utilizada também por todas as demais usinas a jusante numa mesma cascata.

É importante ressaltar que a energia armazenável máxima só será alterada se, por exemplo, houver a entrada em operação de uma nova usina hidroelétrica ao longo do período de estudo.

A energia armazenável máxima é obtida através da seguinte expressão:

$$\overline{EA}(R) = c_1 \sum_{i \in R} \left[\left(\overline{vol}(i) - \underline{vol}(i) \right) \cdot \sum_{j \in J(i)} \rho(j) h_l(j) \right] \tag{12}$$

Onde:

- $R \rightarrow$ Reservatório equivalente;
- $c_1 \rightarrow$ Constante para conversão de unidade de grandezas hidráulicas para grandezas energéticas;
- $\overline{vol}(.) \rightarrow$ Volume máximo do reservatório da usina "*i*";
- $\underline{vol}(.) \rightarrow$ Volume mínimo do reservatório da usina "*i*";
- $J(.) \rightarrow$ Conjunto dos reservatórios situados a jusante de uma usina;
- $\rho(.) \rightarrow$ Produtibilidade da usina;
- $h_l(.) \rightarrow$ Altura da queda líquida da usina.

2.5.2 Energia Afluente

Para as usinas com reservatório de regularização, as vazões afluentes são convertidas em energias afluentes. O valor da energia controlável de um subsistema é obtido através da soma das energias controláveis de todos os reservatórios existentes no subsistema.

Com relação às usinas a fio d'água, as energias a fio d'água são somadas para se obter a energia controlável a fio d'água do subsistema. Somando-se estas duas energias, controláveis e fio d'água, são obtidos os valores das energias afluentes ao subsistema equivalente.

2.5.3 Energia Controlável

A energia controlável é obtida através da soma da afluência natural a cada reservatório multiplicada pela sua produtibilidade média equivalente, somada às produtibilidades das usinas a fio d'água a jusante até encontrar a próxima usina com reservatório. A seguir está amostrada a expressão utilizada no cálculo da energia controlável no mês *k*:

$$EC(R,k) = c_2 \sum_{i \in R} Q_a(i,k) \left(\rho(i) h_l(i) + \sum_{j \in FioJ(i)} \rho(j) h_l(j) \right) \tag{13}$$

Onde:

- $R \rightarrow$ Reservatório equivalente;
- $c_2 \rightarrow$ Constante para conversão de unidade de grandezas hidráulicas para grandezas energéticas;
- $Q_a(.,.) \rightarrow$ Vazão afluente da usina num determinado tempo;
- $\rho(.) \rightarrow$ Produtibilidade da usina;
- $h_l(.) \rightarrow$ Altura da queda líquida da usina;
- $FioJ(.) \rightarrow$ Conjunto das usinas a fio d'água situadas a jusante de uma usina.

2.5.4 Energia a Fio d'água

A energia a fio d'água corresponde às afluências que chegam às usinas a fio d'água. O valor desta afluência é obtido através da afluência natural, descontada as afluências naturais às usinas de reservatório imediatamente a montante. Além disto, existe a limitação da máxima vazão que pode ser transformada em energia pela usina, que é a vazão de engolimento máximo das turbinas. Desta forma, a energia a fio d'água, no mês *k*, é dada por:

$$EFIO(R,k) = c_2 \sum_{j \in Fio(R)} min\left[\left(Q_{max}(j) - \sum_{m \in M(j)} Q_{min}(m)\right), \left(Q_a(j,k) - \sum_{m \in M(j)} Q_a(m,k)\right)\right] \rho(j) h_l(j) \quad (14)$$

Onde:

- $R \rightarrow$ Reservatório equivalente;
- $c_2 \rightarrow$ Constante para conversão de unidade de grandezas hidráulicas para grandezas energéticas;
- $Fio(.) \rightarrow$ Conjunto das usinas a fio d'água do reservatório equivalente;
- $Q_{max}(.) \rightarrow$ Vazão máxima da usina;
- $M(.) \rightarrow$ Conjunto das usinas situadas a montante de uma usina;
- $Q_{min}(.) \rightarrow$ Vazão mínima da usina;
- $Q_a(.,.) \rightarrow$ Vazão afluente da usina num determinado tempo;
- $\rho(.) \rightarrow$ Produtibilidade da usina;
- $h_l(.) \rightarrow$ Altura da queda líquida da usina.

2.5.5 Energia de Vazão Mínima

A energia de vazão mínima depende apenas do conjunto de usinas que foram utilizadas no cálculo do subsistema equivalente. Seu valor máximo é calculado através da multiplicação da descarga mínima obrigatória de cada usina com reservatório pela soma de duas parcelas: a primeira corresponde à multiplicação da produtibilidade pela altura de queda líquida máxima da usina; a segunda corresponde ao somatório do produto da produtibilidade com as alturas de queda líquida de todas as usinas a fio

d'água existentes entre este reservatório e o próximo reservatório a jusante. A expressão para o cálculo do valor máximo de energia de vazão mínima, no mês *k*, é dada por:

$$\overline{EVMIN}(R,k) = c_2 \sum_{i \in R} Q_{min}(i) \left(\rho(i) \overline{h_l}(i) + \sum_{j \in Fio(i)} \rho(j) h_l(j) \right) \quad (15)$$

Onde:

- $R \rightarrow$ Reservatório equivalente;
- $c_2 \rightarrow$ Constante para conversão de unidade de grandezas hidráulicas para grandezas energéticas;
- $Q_{min}(.) \rightarrow$ Vazão mínima da usina;
- $\rho(.) \rightarrow$ Produtibilidade da usina;
- $\overline{h_l}(.) \rightarrow$ Altura da queda líquida máxima da usina;
- $Fio(.) \rightarrow$ Conjunto das usinas a fio d'água do reservatório equivalente;
- $h_l(.) \rightarrow$ Altura da queda líquida da usina.

Os valores médios e mínimos da energia de vazão mínima, respectivamente $EVMIN_{med}$ e $\underline{EVMIN}$, são obtidos de forma similar, substituindo-se a altura de queda líquida máxima pelas alturas de queda correspondentes a um armazenamento de metade do volume útil ($h_{lmed}(i)$) e ao nível mínimo operativo ($h_{lmin}(i)$). Relacionando estes três valores aos valores de energia armazenada (máximo, médio e mínimo), têm-se três pontos a partir dos quais se ajusta um polinômio de segundo grau, obtendo-se a curva de energia de vazão mínima em função da energia armazenada no mês.

2.5.6 Energia Evaporada

A energia evaporada se relaciona com a energia armazenada através de uma função de segundo grau, ajustada aos pontos de mínimo, médio e máximo. Seu valor máximo é calculado através da multiplicação do coeficiente de evaporação de cada reservatório pela área correspondente à altura máxima e pelo produto da produtibilidade com a altura de queda líquida máxima de todas as usinas existentes a jusante. A expressão para cálculo do valor máximo mensal de energia evaporada está mostrada a seguir.

$$\overline{EVAP}(R) = c_3 \sum_{i \in R} \left(e(i) A_{max}(i) \sum_{j \in J(i)} \rho(j) \overline{h_l}(j) \right) \quad (16)$$

Onde:

- $R \rightarrow$ Reservatório equivalente;
- $c_3 \rightarrow$ Constante para conversão de unidade de grandezas hidráulicas para grandezas energéticas;
- $e(.) \rightarrow$ Coeficiente de evaporação da usina;
- $A_{max}(.) \rightarrow$ Área máxima do reservatório da usina;
- $J(.) \rightarrow$ Conjunto dos reservatórios situados a jusante de uma usina;
- $\rho(.) \rightarrow$ Produtibilidade da usina;
- $\overline{h_l}(.) \rightarrow$ Altura da queda líquida máxima da usina.

Os valores mínimos e médios são obtidos da mesma forma, substituindo os valores da área correspondente à altura máxima e a altura máxima pelos respectivos valores mínimos e médios. De posse destes três pontos e com os valores máximo, médio e mínimo da energia armazenada, ajusta-se um polinômio de segundo grau que relaciona a energia evaporada com a energia armazenada.

2.5.7 Geração Hidráulica Máxima

A geração hidráulica máxima, $\overline{GH}$, depende apenas das usinas que foram equivalentadas no subsistema *R*. Desta forma, para cada mês do estudo, são calculados três valores de geração hidráulica máxima: o primeiro corresponde à queda líquida considerando o reservatório no volume mínimo; o segundo corresponde à queda líquida considerando o volume do reservatório em 65% do volume útil; o terceiro corresponde à queda líquida considerando o volume máximo do reservatório. Relacionando estes três valores aos valores de energia armazenada (máximo, médio e mínimo), têm-se três pontos a partir dos quais se ajusta um polinômio de segundo grau, obtendo-se a curva de geração hidráulica máxima em função da energia armazenada num dado período.

$$\overline{GH}(R) = c_4 \sum_{i \in (R+Fio)} (1 - teif(i))(1 - ip(i)) \sum_{j=1}^{NoCjMaq(i)} NoMaq(i,j) PotEf(i,j) \min\left[1, \left(\frac{\overline{h_l}(i)}{h_{nom}(i,j)} \right)^{turb(i)} \right] \quad (17)$$

Onde:

- $R \rightarrow$ Reservatório equivalente;
- $c_4 \rightarrow$ Constante para conversão de unidade de grandezas hidráulicas para grandezas energéticas;
- $R+Fio \rightarrow$ Conjunto de usinas com reservatório e a fio d'água;
- $teif(.) \rightarrow$ Taxa média de indisponibilidade forçada da usina;
- $ip(.) \rightarrow$ Taxa média de indisponibilidade programada da usina;
- $NoCjMaq(.) \rightarrow$ Número de conjunto de máquinas da usina;
- $NoMaq(.,.) \rightarrow$ Número de máquinas de cada conjunto existente na usina;
- $PotEf(.,.) \rightarrow$ Potência efetiva de cada conjunto de máquinas existente na usina;
- $\overline{h_l}(.) \rightarrow$ Altura da queda líquida máxima da usina.
- $h_{nom}(.,.) \rightarrow$ Queda nominal de cada conjunto de máquinas existente na usina;
- $turb(.) \rightarrow$ Constante igual a 1,5, se a turbina da usina for do tipo Francis ou Pelton, ou 1,2, se for do tipo Kaplan.

2.5.8 Geração de Pequenas Usinas

A geração de pequenas usinas corresponde à soma da energia disponível de todas as pequenas usinas, que não foram levadas em consideração no sistema equivalente.

2.5.9 Geração de Usinas Submotorizadas

A geração de usinas submotorizadas é a energia disponível nas novas usinas de reservatórios de acumulação durante o período de motorização, até ser completada a sua potência total, ou seja, até que todas as máquinas estejam operando normalmente.

2.5.10 Energia de Enchimento de Volume Morto

É a energia das vazões afluentes necessárias para o enchimento do volume morto das novas usinas.

2.5.11 Geração de Usinas Térmicas

As gerações das usinas térmicas são levadas em consideração juntamente com o seu custo de produção, obedecendo aos seus limites máximos e mínimos operativos. Podem ser agrupadas, desde que o conjunto tenha características iguais de custo e operação.

2.5.12 Intercâmbios

Consiste nas energias trocadas com outros subsistemas. Deve obedecer aos limites máximos e mínimos operativos do carregamento das linhas de transmissão que interligam os submercados.

2.5.13 Mercado Consumidor

O mercado consumidor é a energia a ser atendida pelo sistema. Para representar as eventuais situações de *deficit*, que é a parcela não atendida do mercado consumidor, o mesmo deverá ser levado em consideração no problema de planejamento através de um custo.

2.6 Conceitos de Função de Custo Presente e Função de Custo Futuro

No planejamento da operação de sistemas hidrotérmicos deve-se evitar que as decisões de se utilizar ou não a energia das usinas hidráulicas causem custo elevado da operação futura ou por conta da necessidade de geração térmica, ou da impossibilidade de atendimento da demanda ou do vertimento de água, que seria desperdiçar energia armazenada. Deve-se, portanto, dosar a geração hidráulica, que fornece um benefício imediato do seu uso gratuito e um benefício futuro, por conta da energia armazenada nos reservatórios.

Na Figura 7 estão apresentados os gráficos das funções de custo futuro e presente.

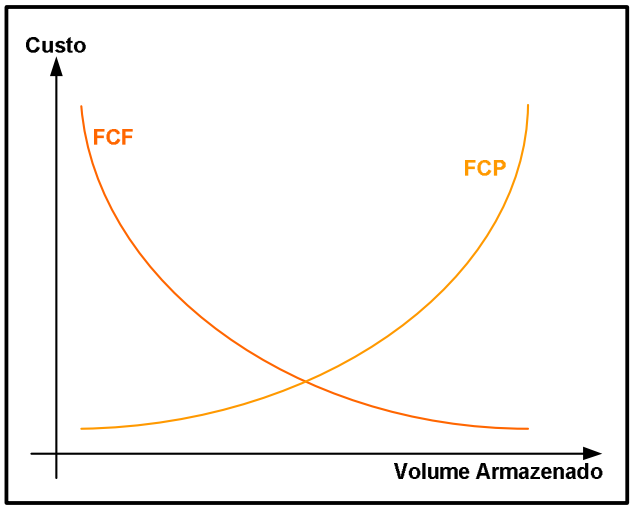

Figura 7 – Funções de Custo Futuro e Presente

A função de custo presente (FCP) mede o custo da geração térmica no início do estágio presente. Este custo é baixo quando da utilização de bastante recurso hidráulico, o que diminui o volume armazenado nos reservatórios.

A função de custo futuro (FCF) mede o custo da geração térmica e do deficit de suprimento da demanda do final do estágio presente até o final do horizonte de planejamento. Ela tem um comportamento oposto ao da FCP, uma vez que, quando o volume armazenado está alto, existe uma boa quantidade de energia hidráulica, não necessitando de geração térmica, o que implica num custo baixo de operação. Este custo aumenta com a diminuição da energia armazenada por conta da necessidade de mais geração de energia térmica para suprir a demanda.

O uso ótimo da água correponde ao ponto de mínimo da função obtida da soma dos custos presente e futuro, conforme está mostrado na Figura 8.

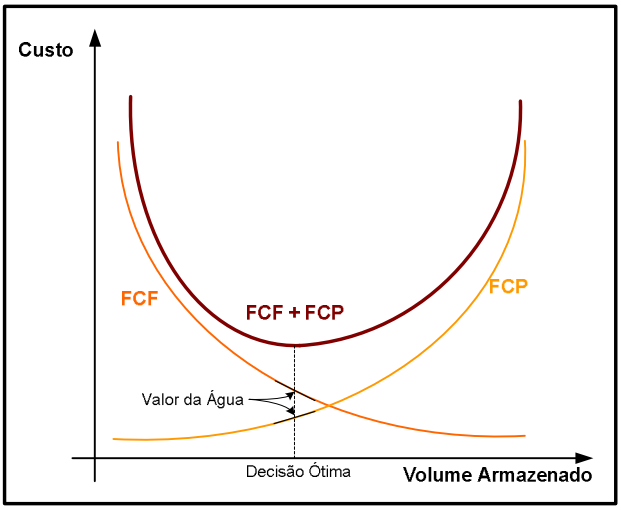

Figura 8 – Decisão Ótima do Uso da Água

O ponto de decisão ótima do uso da água também corresponde aos pontos em as derivadas das duas funções, FCP e FCF, são iguais em módulo. Estas derivadas também são chamadas de valores da água. Logo, o uso da água das usinas hidrelétricas têm um valor, que está associado à economia da geração térmica presente ou futura.

2.7 Formulação do Problema do Planejamento da Operação Hidrotérmica

O planejamento da operação energética ótima de um sistema de geração hidrotérmico tem como objetivo determinar, a cada período, uma meta de operação para cada unidade geradora do sistema que minimize o custo esperado de operação ao longo do período de planejamento. O custo de operação é composto de custo de combustível das unidades termoelétricas e penalidades por eventuais não atendimentos da demanda de energia.

Desta forma, o problema de operação ótima de um sistema hidrotérmico consiste em determinar uma estratégia de operação que a cada estágio do período de planejamento, conhecido o estado do sistema no início do estágio, forneça as metas de geração de cada unidade geradora. Esta estratégia deve minimizar o valor esperado do custo de operação ao longo de todo o período de estudo, composto de custos de combustível, acrescido das penalizações por eventuais falhas no atendimento de energia. O estado do sistema é composto por variáveis que podem influir no resultado da operação, sendo considerados o nível de armazenamento no início do estágio e as

energias afluentes dos estágios anteriores. Logo, para cada estágio, deseja-se minimizar a soma do custo presente, representado pela soma do custo dos combustíveis consumidos pelas usinas termoelétricas para a geração de energia com o custo da interrupção do fornecimento de energia (custo de *deficit*), com o custo futuro de operação, representado pelo impacto futuro de uma decisão tomada no período corrente, que é o custo de operação do estágio seguinte ao corrente até o final do horizonte do estudo.

As equações representativas da FCF, também chamados de cortes de Benders serão explicadas no item 2.9.1, referente à programação dinâmica dual estocástica.

O problema de operação hidrotérmica pode ser representado da seguinte forma:

Min Custo Geração Térmica + Custo de *deficit* + Custo Futuro

Sujeito a

- Equações de balanço hídrico
- Equações de atendimento à demanda
- Equações de restrições representativas da função de custo futuro
- Equações de restrições de geração hidráulica máxima controlável
- Equações de nós

Respeitados os seguintes limites

1. Intercâmbios entre submercados
2. Gerações térmicas mínimas e máximas das usinas
3. Energias armazenadas mínimas e máximas nos finais dos estágios

1. Função Objetivo

$$\alpha^k = Min\left\{\sum_R \left[C_T\left(GT(R,k)\right) + C_D\left(DEF(R,k)\right)\right] + \frac{1}{1+\beta}\alpha^{k+1}\right\} \tag{18}$$

2. Sujeito às Equações de Balanço Hídrico

$$EA(R,k+1) = EA(R,k) + EC(R,k) - GH(R,k) - EVERT(R,k) - EVAP(R,k) - EVMOR(R,k) \tag{19}$$

3. Sujeito às Equações de Atendimento à Demanda

$$GH(R,k) + GT(R,k) + \sum_{s \neq R}\left(INT(s,R,k) - INT(R,s,k)\right) = DEM(R,k) - DEF(R,k) \tag{20}$$

4. Sujeito às Equações de Restrições Representativas da Função de Custo Futuro

$$\begin{aligned}
&\alpha^{k+1} - \sum_R \pi EA_1(R,k+1)\cdot EA(R,k+1) \geq \delta_1^{k+1} \qquad (21)\\
&\alpha^{k+1} - \sum_R \pi EA_2(R,k+1)\cdot EA(R,k+1) \geq \delta_2^{k+1}\\
&\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots\\
&\alpha^{k+1} - \sum_R \pi EA_c(R,k+1)\cdot EA(R,k+1) \geq \delta_c^{k+1}
\end{aligned}$$

5. Sujeito às Equações de Restrições de Geração Hidráulica Máxima Controlável

$$\underline{GH}(R) \leq GH(R,k) \leq \overline{GH}(R) \tag{22}$$

6. Sujeito às Equações de nós

$$\sum_{\forall s \neq R} INT(s,R,k) - \sum_{\forall s \neq R} INT(R,s,k) = 0 \tag{23}$$

7. Limites de Intercâmbios entre Submercados

$$\underline{INT}(s,R) \leq INT(s,R,k) \leq \overline{INT}(s,R) \tag{24}$$

8. Limites de Gerações Térmicas Mínimas e Máximas das Usinas

$$\underline{GT}(R) \leq GT(R,k) \leq \overline{GT}(R) \tag{25}$$

9. Limites de Energias Armazenadas Mínimas e Máximas nos Finais dos Estágios

$$\underline{EA}(R,k+1) \leq EA(R,k+1) \leq \overline{EA}(R,k+1) \tag{26}$$

Onde:

$k \rightarrow$	Período.
$R \rightarrow$	Subsistema.
$\alpha \rightarrow$	Custo de operação do período.
$C_T \rightarrow$	Função que calcula o custo de geração térmica.
$GT \rightarrow$	Geração das usinas térmicas de um subsistema num determinado período. Seu valor máximo é representado por $\overline{GT}$ e seu valor mínimo é representado por $\underline{GT}$.
$C_D \rightarrow$	Função que calcula o custo do não atendimento da demanda.
$DEF \rightarrow$	Demanda não atendida (Déficit) de um subsistema num determinado período.
$\beta \rightarrow$	Taxa de desconto de um valor num período futuro para o valor presente.
$EA \rightarrow$	Energia armazenada de um subsistema num determinado período. Seu valor máximo é representado por $\overline{EA}$ e seu valor mínimo por $\underline{EA}$.
$EC \rightarrow$	Energia controlável de um subsistema num determinado período.
$GH \rightarrow$	Geração das usinas hidráulicas de um subsistema num determinado período. Seu valor máximo é representado por $\overline{GH}$.
$EVERT \rightarrow$	Energia perdida através de vertimento de um subsistema num determinado período.
$EVAP \rightarrow$	Energia perdida através de evaporação de um subsistema num determinado período.
$EVMOR \rightarrow$	Energia perdido por conta do enchimento de volume morto de um subsistema num determinado período.
$INT \rightarrow$	Intercâmbio de energia entre dois subsistemas num determinado período. Seu valor máximo é indicado por $\overline{INT}$.
$DEM \rightarrow$	Demanda de mercado a ser suprida num subsistema num determinado período.
$\pi EA \rightarrow$	Derivada da função objetivo em relação à energia armazenada de um subsistema num determinado período. Este parâmetro é função do número do corte de Benders.
$\delta \rightarrow$	Termo constante da restrição linear do corte de Benders
$s \rightarrow$	Subsistema.
$c \rightarrow$	Corte de Benders.

2.8 Programação Dinâmica Estocástica

O problema de planejamento da operação consiste em se determinar a quantidade de geração hidráulica que será gerada em cada um dos períodos que compõem o horizonte de planejamento, de forma a minimizar o custo total de operação.

Uma grande dificuldade deste problema é que a quantidade disponível de água, combustível "gratuito" utilizado pelas usinas hidráulicas, não é conhecida a priori, uma

vez que as afluências que ocorrerão ao longo do período de planejamento são desconhecidas, o que caracteriza a estocasticidade do problema. O que é conhecido é a distribuição da probabilidade das afluências futuras, condicionadas às afluências dos períodos passados. Uma ferramenta bastante utilizada para se obter a solução ótima deste problema é a programação dinâmica estocástica (PDE).

A PDE, formulada inicialmente por Bellman [119], consiste em dividir um problema em etapas, determinando a melhor decisão para cada uma delas, de acordo com o estado em que se encontra o sistema. A decisão ótima é baseada no conhecimento prévio de todas as possibilidades futuras e suas consequências. Para que a decisão atenda ao princípio da otimalidade de Bellman, basta resolver as etapas no sentido inverso, do final do período para o início, tomando sempre a decisão ótima em cada uma delas.

Nesta formulação, o custo total de operação é estimado pela soma do custo presente, devido à decisão tomada no período corrente, com o valor atual do custo futuro, que depende das decisões ótimas de todos os períodos subsequentes. As variáveis de estado seriam a energia armazenada no reservatório da usina hidráulica e a energia da vazão afluente e a variável de decisão seria a vazão turbinada. O espaço de estado é então discretizado entre os valores mínimo e máximo do volume útil do reservatório e diversos valores de afluências, com suas respectivas probabilidades de ocorrência. As decisões possíveis são também discretizadas entre as vazões turbinadas mínima e máxima. Para cada período, o estado final do armazenamento depende do estado inicial do armazenamento, da vazão turbinada e da afluência. Esta formulação da PDE está mostrada na Figura 9.

Para cada período do horizonte de planejamento, deve-se avaliar o custo de todas as decisões de energia turbinada, levando-se em consideração todos os possíveis valores de energia afluente e suas probabilidades e todos os possíveis estados de armazenamento. Neste ponto, pode-se perceber o grande problema da programação dinâmica, que é a chamada maldição da dimensionalidade. Supondo que o volume de armazenamento esteja discretizado em 10 estados e as vazões afluentes possuam 10 estados diferentes, a quantidade de estados a serem avaliados serão 100. A inclusão de um segundo reservatório, implica que a quantidade de estados aumente para 100^2, ou seja, 10.000 estados. Para cinco reservatórios seriam 10.000.000.000 estados. Esta

característica inibe a representação de vários reservatórios, uma vez que o esforço computacional aumenta exponencialmente. Por conta disto, outras técnicas tiveram que ser desenvolvidas para a solução do problema do planejamento de sistemas hidrotérmicos.

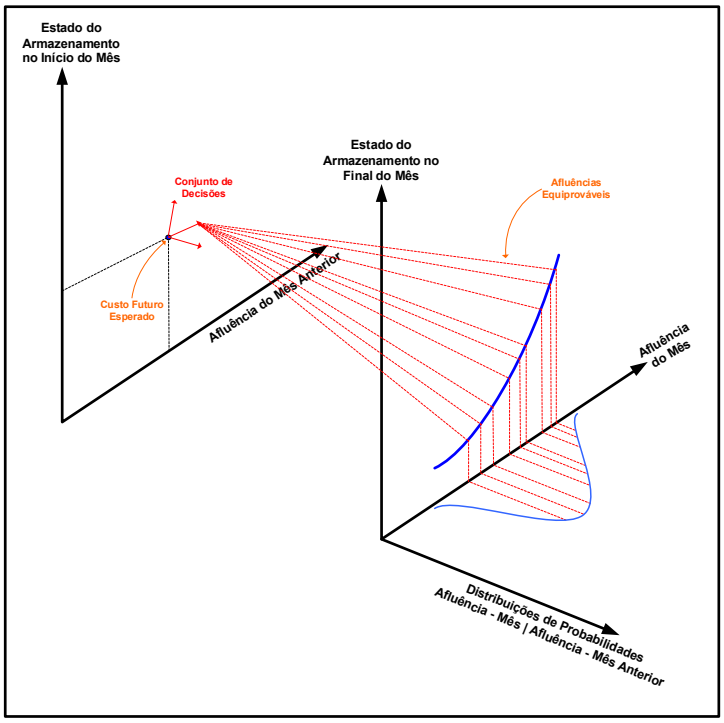

Figura 9 – Formulação da Programação Dinâmica Estocástica

2.9 Programação Dinâmica Dual Estocástica

O principal objetivo da programação dinâmica dual estocástica (PDDE) é resolver um problema de otimização evitando a explosão combinatória apresentada na PDE. A PDDE é uma metodologia baseada na construção analítica da função de custo futuro utilizando a decomposição de Benders.

2.9.1 Programação Dinâmica Dual Determinística

De forma a facilitar o entendimento do método utilizado na PDDE, será apresentado inicialmente a aplicação da metodologia num problema determinístico, onde será aplicada a programação dinâmica dual determinística (PDDD), estendendo

posteriormente os conceitos para um problema estocástico. O desenvolvimento desta metodologia também pode ser visto em várias referências, destacando-se [120], [113] e [12].

Considerando o seguinte problema de programação linear.

$$f = \underset{z_1, z_2}{Min} \; c_1 z_1 + c_2 z_2$$

$$\text{Sujeito a} \begin{cases} A_1 z_1 & \geq b_1 \\ E_1 z_1 + A_2 z_2 & \geq b_2 \\ z_1 & \geq 0 \\ z_2 & \geq 0 \end{cases} \tag{27}$$

Este problema pode ser resolvido através da solução de um processo de decisão de dois estágios. Inicialmente escolhe-se um conjunto viável z_1^v que satisfaça as condições $A_1 z_1^v \geq b_1$ e $z_1^v \geq 0$. O segundo estágio consiste em resolver o seguinte problema.

$$f = \underset{z_2}{Min} \; c_2 z_2$$

$$\text{Sujeito a} \begin{cases} A_2 z_2 \geq b_2 - E_1 z_1^v \\ z_2 \quad \geq 0 \end{cases} \tag{28}$$

Como o conjunto z_1^v é conhecido, ele passa para o lado direito da restrição do problema.

O objetivo deste processo é encontrar uma solução que minimize a soma das funções objetivos do primeiro e do segundo estágios, sendo que os vetores z_1 e z_2 representam as energias armazenadas nos volumes finais dos reservatórios, energias turbinadas e vertidas, energia gerada pelas usinas termelétricas etc., para os respectivos estágios de solução.

Desta forma, pode-se reescrever o problema do primeiro estágio (27) da seguinte forma.

$$f = \underset{z_1}{Min}\ \ c_1 z_1 + v(z_1)$$

$$\text{Sujeito a } \begin{cases} A_1 z_1 & \geq b_1 \\ z_1 & \geq 0 \end{cases} \tag{29}$$

onde a parcela $c_1 z_1$ corresponde ao custo imediato referente ao primeiro estágio e a parcela $v(z_1)$ corresponde ao custo futuro da decisão z_1. Esta parcela é definida da seguinte forma.

$$v(z_1) = \underset{z_2}{Min}\ \ c_2 z_2$$

$$\text{Sujeito a } \begin{cases} A_2 z_2 \geq b_2 - E_1 z_1 \\ z_2 \quad \geq 0 \end{cases} \tag{30}$$

A parcela $v(z_1)$ fornece o custo ótimo do problema do segundo estágio, a partir da decisão z_1 tomada no problema do primeiro estágio. O problema é que a representação explícita da função $v(z_1)$ não é conhecida e, conforme será visto a seguir, a programação dinâmica dual constrói iterativamente uma aproximação linear por partes que representa o valor desta função no problema de primeiro estágio definido em (29). O processo de construção da função $v(z_1)$, também conhecido como o princípio da decomposição de Benders, será descrito a seguir.

Partindo-se da afirmação que para todo problema de programação linear existe outro problema dual associado, pode-se representar o problema original, chamado de primal, através das variáveis duais, conforme está mostrado a seguir.

$$v(z_1) = \underset{\pi}{Max}\ \ \pi(b_2 - E_1 z_1)$$

$$\text{Sujeito a } \begin{cases} \pi A_2 \leq c_2 \\ \pi \quad \geq 0 \end{cases} \tag{31}$$

onde π representa o conjunto de variáveis duais associadas às restrições do problema do segundo estágio.

Os valores das funções objetivos dos problemas representados pelas equações (30) e (31) coincidem nas respectivas soluções ótimas, logo, ambos podem ser

utilizados para encontrar o valor de $v(z_1)$, porém a região viável do problema dual, definida pelo conjunto de suas restrições, não depende dos valores de z_1.

A região viável do problema dual consiste num poliedro convexo formado pelos pontos extremos estabelecidos pelo conjunto de todas as restrições que atendam as condições definidas em (31). Como a solução ótima se situa num destes pontos extremos, o problema poderia ser resolvido por enumeração. Convém ressaltar que quanto maior for a quantidade de restrições, mais próxima a função linearizada estará da função real, implicando numa solução ótima mais próxima da solução ótima real. O problema é que a determinação de todos os pontos extremos implicaria na construção completa da função $v(z_1)$ e este processo é de elevado tempo computacional. A metodologia da PDDD objetiva determinar um subconjunto de pontos extremos da função objetivo de forma a obter o valor ótimo com uma boa aproximação, utilizando um tempo computacional significativamente menor.

Supondo que ao longo do processo de construção da função $v(z_1)$, problema (31), um subconjunto $\pi = \{\pi^1, \pi^2, \pi^3, \ldots, \pi^{n-2}, \pi^{n-1}, \pi^n\}$ com n pontos extremos tenha sido obtido. Como já foi dito anteriormente, a solução ótima se situa num destes pontos e o problema poderia ser resolvido por enumeração, conforme está mostrado em (32).

$$v(z_1) = \underset{\pi}{Max} \{\pi^i (b_2 - E_1 z_1), i = 1,2,\ldots n\} \tag{32}$$

Este problema pode ser reescrito como um problema de programação linear, conforme pode ser visto a seguir.

$$v(z_1) = \underset{v}{Min} \;\; v$$

$$\text{Sujeito a} \begin{cases} v \geq \pi^1 (b_2 - E_1 z_1) \\ v \geq \pi^2 (b_2 - E_1 z_1) \\ v \geq \pi^3 (b_2 - E_1 z_1) \\ \vdots \quad \vdots \qquad \vdots \\ v \geq \pi^{n-2} (b_2 - E_1 z_1) \\ v \geq \pi^{n-1} (b_2 - E_1 z_1) \\ v \geq \pi^n (b_2 - E_1 z_1) \end{cases} \tag{33}$$

Como o problema é de minimização e as restrições impõem uma fronteira mínima à região viável das possíveis soluções, a solução ótima certamente estará em um dos pontos desta fronteira.

A interpretação geométrica do problema (33) está apresentada na Figura 10, para um exemplo com *n* igual a 5 restrições. Pode-se observar que a fronteira da função $\upsilon(z_1)$, onde está situada a solução ótima do problema, é constituída de vários segmentos de reta, onde as inclinações das mesmas correspondem aos valores de $\pi^1(b_2 - E_1 z_1)$ até $\pi^5(b_2 - E_1 z_1)$.

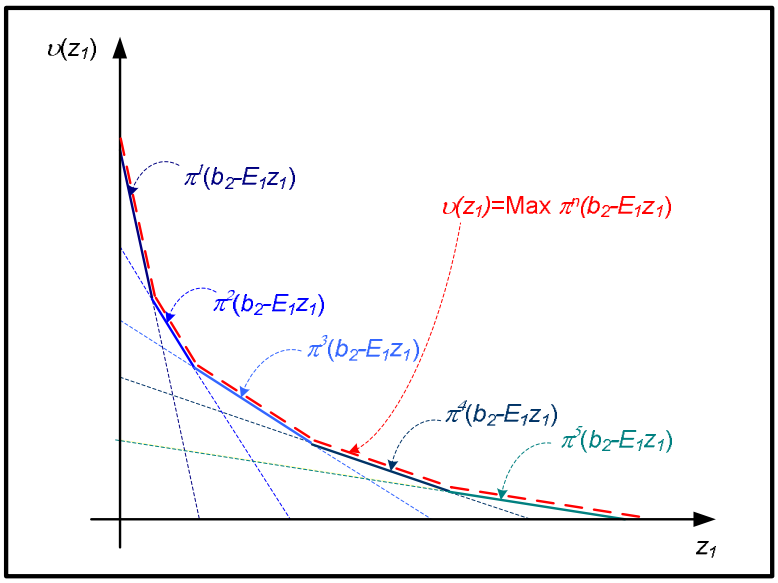

Figura 10 – Interpretação Geométrica da Montagem da Função de Custo Futuro

Conforme se aumenta os conjuntos de soluções, a quantidade de restrições vai aumentando, tornando o problema cada vez mais restrito, conforme mostrado em (34), e menos relaxado em relação ao problema real. Logo, com o aumento das restrições a solução ótima fica cada vez mais próxima da solução ótima real.

$$\upsilon(z_1) = \underset{\upsilon}{Min}\ \upsilon$$

$$\begin{array}{ll}
z_1^1 \rightarrow \pi^1 \Rightarrow \upsilon \geq \pi^1(b_2 - E_1 z_1) & \rightarrow \textit{restrição 1} \\
z_1^2 \rightarrow \pi^2 \Rightarrow \upsilon \geq \pi^2(b_2 - E_1 z_1) & \rightarrow \textit{restrição 2} \\
z_1^3 \rightarrow \pi^3 \Rightarrow \upsilon \geq \pi^3(b_2 - E_1 z_1) & \rightarrow \textit{restrição 3} \\
\vdots \quad \vdots \quad \vdots & \vdots \\
z_1^{n-2} \rightarrow \pi^{n-2} \Rightarrow \upsilon \geq \pi^{n-2}(b_2 - E_1 z_1) & \rightarrow \textit{restrição } n-2 \\
z_1^{n-1} \rightarrow \pi^{n-1} \Rightarrow \upsilon \geq \pi^{n-1}(b_2 - E_1 z_1) & \rightarrow \textit{restrição } n-1 \\
z_1^n \rightarrow \pi^n \Rightarrow \upsilon \geq \pi^n(b_2 - E_1 z_1) & \rightarrow \textit{restrição } n
\end{array} \tag{34}$$

Supondo que o problema definido em (30) tenha sido resolvido para um primeiro vetor de z_1^1, ele ficaria com a seguinte forma.

$$\upsilon\left(z_1^1\right) = \underset{z_2}{Min}\ c_2 z_2$$

$$\text{Sujeito a } \begin{cases} A_2 z_2 \geq b_2 - E_1 z_1^1 \\ z_2 \quad \geq 0 \end{cases} \tag{35}$$

A solução do problema dual fornece um conjunto de variáveis duais π^1 associadas às restrições do problema (35). Este conjunto de variáveis corresponde ao ponto extremo ótimo de (35) da solução da função $\upsilon(z_1)$, que, por sua vez, define a precisão da solução ótima do problema. Porém, a inclusão de apenas uma restrição não fornece uma boa aproximação da função $\upsilon(z_1)$ real, implicando numa provável solução ótima longe do valor ótimo real. A solução ótima fica cada vez mais próxima do valor ótimo real quando se obtém, por exemplo, um conjunto de *n* vetores z_1, representados por $\left\{z_1^1, z_1^2, z_1^3, \ldots, z_1^n\right\}$, obtendo-se *n* vetores de variáveis duais $\left\{\pi^1, \pi^2, \pi^3, \ldots, \pi^n\right\}$, onde o ponto extremo ótimo de (35) será um destes elementos do conjunto π, utilizados para a determinação da solução aproximada de $\upsilon(z_1)$, conforme pode ser visto em (36).

$$\underline{f} = \underset{z_1, \upsilon}{Min} \;\; c_1 z_1 + \upsilon$$

$$\text{Sujeito a} \begin{cases} A_1 z_1 \geq b_1 \\ \upsilon \geq \pi^1 \left(b_2 - E_1 z_1 \right) \\ \upsilon \geq \pi^2 \left(b_2 - E_1 z_1 \right) \\ \vdots \\ \upsilon \geq \pi^n \left(b_2 - E_1 z_1 \right) \\ z_1 \geq 0 \end{cases} \tag{36}$$

A solução ótima final é obtida com a convergência do processo iterativo, onde o limite inferior é o valor da solução do problema definido em (36), que é relaxado em relação ao problema definido em (29), já que nem todos os pontos extremos da função $\upsilon(z_1)$ foram levados em consideração. Esta conclusão também pode ser obtida observando-se o gráfico da Figura 10. As novas restrições que se situassem entre a fronteira mínima de solução e os eixos do gráfico não seriam levadas em consideração porque as atuais restrições implicam que essa região não é viável. Já as restrições que surgissem na atual região viável, e fossem ativas na solução ótima, implicariam em valores maiores que o atual, levando-se à conclusão que o valor atual, apesar de não ter a garantia de ser o ótimo, é garantidamente menor que ele.

O limite superior do valor ótimo é obtido através da solução do problema (37), descrito a seguir, com a utilização de z_1 obtido no problema (36).

$$\overline{f} = c_1 z_1 + \underset{z_2}{Min}\, c_2 z_2$$

$$\text{Sujeito a} \begin{cases} A_2 z_2 \geq b_2 - E_1 z_1 \\ z_2 \geq 0 \end{cases} \tag{37}$$

O valor ótimo do problema (37) não tem a garantia de ser o ótimo do problema original, por conta das aproximações do método, mas é uma solução viável. Logo, o valor ótimo real deste problema de minimização deverá ser menor ou igual a esta solução viável, concluindo que a solução ótima do problema (37) é um limite superior para a solução ótima do problema real.

De posse dos limites máximo e mínimo, $\overline{f}$ e $\underline{f}$, respectivamente, deve-se utilizar uma tolerância para verificar se o valor da diferença $\Delta f = \overline{f} - \underline{f}$ é aceitável ou não. Caso a diferença seja maior que a tolerância adotada, deve-se utilizar um novo conjunto de solução z_1 para se determinar outro conjunto de restrições, a serem adicionadas às anteriores, para melhorar a aproximação e diminuir a diferença Δf até que ela seja menor que a tolerância desejada e o problema seja considerado resolvido.

2.9.2 Extensão para o Caso Estocástico

Estendendo-se o problema apresentado em (27), tem-se o seguinte.

$$f = \underset{z_1, z_{21} \, a \, z_{2k}}{Min} \; c_1 z_1 + p_1 c_{21} z_{21} + p_2 c_{22} z_{22} + p_3 c_{23} z_{23} + \cdots + p_{k-1} c_{2(k-1)} z_{2(k-1)} + p_k c_{2k} z_{2k}$$

$$\text{Sujeito a} \begin{cases} A_1 z_1 \geq b_1 \\ E_1 z_1 + A_{21} z_{21} \geq b_{21} \\ E_1 z_1 + A_{22} z_{22} \geq b_{22} \\ E_1 z_1 + A_{23} z_{23} \geq b_{23} \\ \quad \vdots \qquad \vdots \qquad \vdots \\ E_1 z_1 + A_{2(k-1)} z_{2(k-1)} \geq b_{2(k-1)} \\ E_1 z_1 + A_{2k} z_{2k} \geq b_{2k} \\ z_1, z_{21}, z_{22}, \ldots, z_{2k} \geq 0 \end{cases} \tag{38}$$

onde p_1, p_2, p_3, ..., p_{k-1}, p_k são as probabilidades de ocorrência de cada um dos k cenários.

Este problema pode ser decomposto em $k+1$ problemas de programação linear, sendo um de primeiro estágio e k problemas de segundo estágio, um para cada cenário.

Desta forma, tem-se para o problema de primeiro estágio a seguinte formulação.

$$f = \underset{z_1}{Min} \; c_1 z_1 + p_1 v_1(z_1) + p_2 v_2(z_1) + p_3 v_3(z_1) + \cdots + p_k v_k(z_1) \text{ ou}$$

$$f = \underset{z_1}{Min} \; c_1 z_1 + \sum_{j=1}^{k} p_j v_j(z_1) \tag{39}$$

$$\text{Sujeito a } \begin{cases} A_1 z_1 \geq b_1 \\ z_1 \geq 0 \end{cases}$$

Para o problema de segundo estágio referente ao primeiro cenário, tem-se a seguinte formulação.

$$\upsilon_1(z_1) = \underset{z_{21}}{Min} \; c_{21} z_{21}$$

$$\text{Sujeito a } \begin{cases} A_{21} z_{21} \geq b_{21} - E_1 z_1 \\ z_{21} \quad \geq 0 \end{cases} \tag{40}$$

Para o problema de segundo estágio referente ao segundo cenário, tem-se a seguinte formulação.

$$\upsilon_2(z_1) = \underset{z_{22}}{Min} \; c_{22} z_{22}$$

$$\text{Sujeito a } \begin{cases} A_{22} z_{22} \geq b_{22} - E_1 z_1 \\ z_{22} \quad \geq 0 \end{cases} \tag{41}$$

Generalizando-se para os *k* cenários, o problema de segundo estágio fica na seguinte forma genérica.

$$\upsilon_k(z_1) = \underset{z_{2k}}{Min} \; c_{2k} z_{2k}$$

$$\text{Sujeito a } \begin{cases} A_{2k} z_{2k} \geq b_{2k} - E_1 z_1 \\ z_{2k} \quad \geq 0 \end{cases} \tag{42}$$

Neste caso, o objetivo do processo é obter uma solução que minimize a soma do valor ótimo do problema do primeiro estágio com o valor esperado dos valores ótimos dos *k* problemas de segundo estágio. Um diagrama representativo de todo este processo [120] está mostrado na Figura 11.

A solução para o caso estocástico é similar à obtida no caso determinístico, ou seja, a programação dinâmica dual vai construindo iterativamente uma aproximação linear por partes para cada uma das *k* funções υ_k. A cada iteração, novas restrições vão

sendo acrescentadas no problema (39) e novos limites superior e inferior são obtidos. Este processo se repete até que a diferença entre os limites estejam dentro de uma tolerância pré-definida, caso em que se obtém a convergência do processo iterativo.

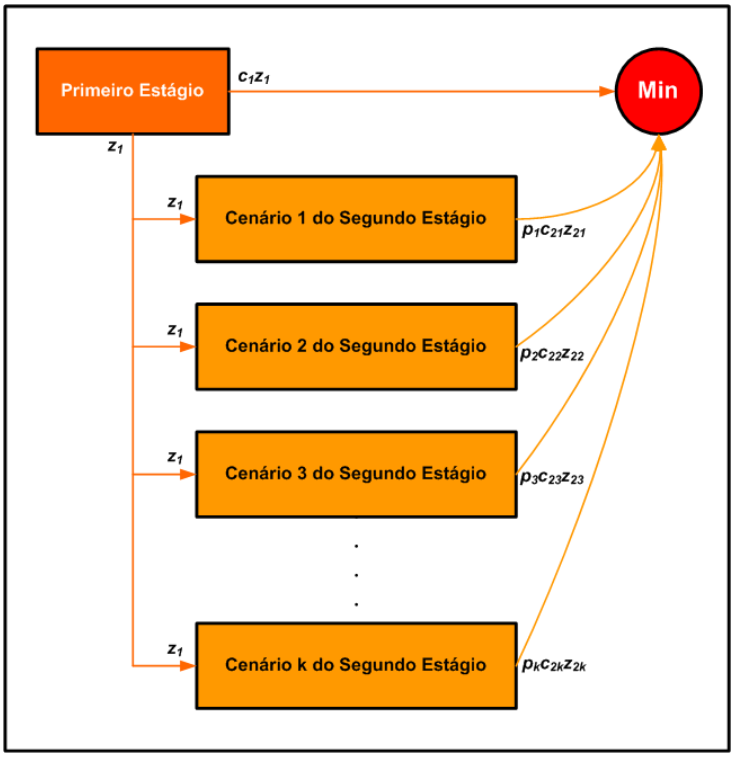

Figura 11 – Processo de Decisão de Dois Estágios do Caso Estocástico

A seguir será apresentado um algoritmo para a solução de problemas de programação estocástica [120].

1. Inicializar o contador de iterações $m = 0$.
2. Determinar o limite inferior para a solução ótima do problema original através da solução do problema a seguir.

$$\underline{f} = \underset{z_1, \upsilon}{Min} \; c_1 z_1 + \sum_{j=1}^{k} p_j \upsilon_j \qquad (43)$$

$$\text{Sujeito a} \begin{cases} A_1 z_1 \geq b_1 \\ \upsilon_1 \geq \pi_1^1 \left(b_{21} - E_1 z_1 \right) \\ \upsilon_2 \geq \pi_2^1 \left(b_{22} - E_1 z_1 \right) \\ \vdots \\ \upsilon_k \geq \pi_k^1 \left(b_{2k} - E_1 z_1 \right) \\ z_1 \geq 0 \end{cases}$$

3. A partir do vetor de solução z_1, obtido no item 2, resolver os problemas de segundo estágio para cada cenário *k*.

$$\upsilon_k(z_1) = \underset{z_{2k}}{Min} \; c_{2k} z_{2k}$$

$$\text{Sujeito a} \begin{cases} A_{2k} z_{2k} \geq b_{2k} - E_1 z_1 \\ z_{2k} \quad \geq 0 \end{cases} \tag{44}$$

4. Armazenar os vetores de variáveis duais associados às restrições dos problemas de segundo estágio para cada cenário *k*.
5. Determinar o limite superior para a solução ótima do problema original através da operação $\overline{f} = c_1 z_1^1 + \sum_{j=1}^{k} p_j \upsilon_j(z_1)$.
6. Verificar se $\Delta f = \overline{f} - \underline{f} \leq \varepsilon$. Se afirmativo, pare o processo. Caso contrário siga para o próximo passo.
7. Incrementar o contador de iterações *m = m + 1*.
8. Voltar para o problema do passo 2, atualizando as restrições com os valores das variáveis duais armazenadas no passo 4.

Capítulo 3.
Computação Paralela

O objetivo da computação paralela [1], [121], [122], [123], [124] e [125] é a redução do tempo computacional de uma aplicação. Esta redução é possível através da identificação das chamadas tarefas concorrentes, que são instruções independentes entre si e que podem ser processadas ao mesmo tempo. Cada vez mais este tipo de computação está se tornando a solução para engenheiros e cientistas resolverem alguns problemas que seriam "insolúveis" com a utilização do processamento convencional.

A computação paralela abrange três grandes itens da computação, que são os seguintes: arquiteturas de computadores (*hardware*); os modelos de programação paralela (*software*); e a implementação de programas com processamento paralelo (aplicações e algoritmos). Nos itens a seguir serão comentadas as características destes assuntos.

3.1 Arquiteturas de Computadores (*hardware*)

Com relação às arquiteturas de computadores, nos itens a seguir serão comentados os tipos de arquiteturas paralelas, os tipos de máquinas utilizadas para processamento paralelo e as características principais dos *clusters* de computadores.

3.1.1 Arquiteturas Paralelas

Talvez a classificação das arquiteturas paralelas mais utilizada ao longo dos anos seja a que foi proposta for Flynn. Esta classificação, conhecida por taxonomia de Flynn [126] (*Flynn´s taxonomy*), é baseada nos fluxos de instruções e dados dos processadores e apresenta quatro classes de processamento.

(1) SISD (*Single Instruction Single Data*) → O termo *Single Instruction* significa que existe apenas uma via de instrução, ou seja, o fluxo de instruções para o processador é único, assim como o fluxo de dados (*Single Data*). Nesta categoria estão os processadores convencionais, não paralelos, uma vez que as instruções e os dados são tratados de forma seqüencial, conforme está exemplificado na Figura 12 [125]. Convém informar que em muitos deles, por motivos de aumento de desempenho, possuem estruturas de *pipeline*, que atuam como se fossem uma linha de montagem.

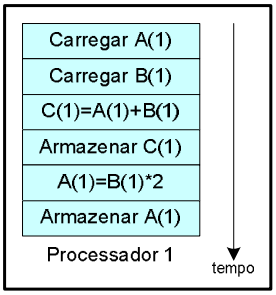

Figura 12 – Arquitetura de Processamento SISD

(2) SIMD (*Single Instruction Multiple Data*) → Nesta arquitetura tem-se um fluxo único de instruções e um fluxo múltiplo de dados. Todas as unidades de processamento executam a mesma instrução, porém em diferentes conjuntos de dados, permitindo, portanto, um tipo de execução em paralelo, conforme está mostrado na Figura 13. É indicada para processamento de problemas especializados, caracterizados por um alto grau de regularidade, tais como, processamento de imagens e gráficos. Os processadores vetoriais estão normalmente incluídos nesta classificação.

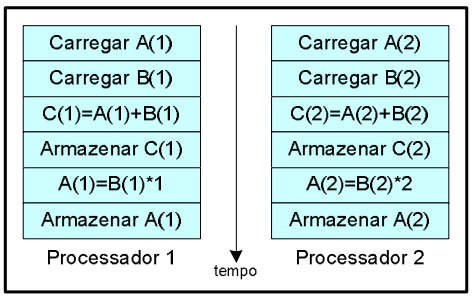

Figura 13 – Arquitetura de Processamento SIMD

(3) MISD (*Multiple Instructions Single Data*) → Esta arquitetura possui um fluxo múltiplo de instruções e somente um fluxo de dados, ou seja, os vários processadores desta arquitetura executam operações distintas num mesmo conjunto de dados, conforme está mostrado na Figura 14. Nunca existiram muitos exemplos de computadores nesta classe, sendo que alguns autores nem a consideram por não ser significativa. Um exemplo citado [125] é do computador experimental "Carnegie-Mellon C.mmp", de 1971 e um exemplo de operação que caberia neste tipo de arquitetura seria a execução de vários algoritmos de criptografia com o objetivo de quebrar uma mensagem criptografada [125].

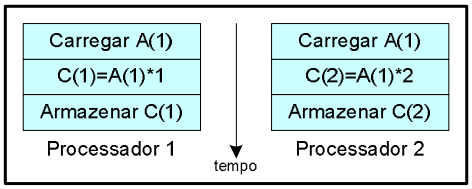

Figura 14 – Arquitetura de Processamento MISD

(4) MIMD (*Multiple Instructions Multiple Data*) → Esta arquitetura possui fluxos múltiplos tanto para instruções quanto para dados. Isso significa que os processadores podem realizar diferentes instruções em diferentes conjuntos de dados, conforme está exemplificado na Figura 15. A grande maioria dos computadores modernos cai nesta classificação, tais como: computadores do tipo SMP; computadores pessoais com processadores de vários núcleos; *clusters* etc.

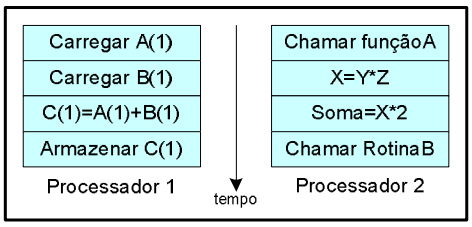

Figura 15 – Arquitetura de Processamento MIMD

Atualmente existe outra importante classificação que é o tipo de arquitetura de memória utilizada pelo computador. Nesta classificação existem três tipos possíveis, conforme está descrito a seguir.

(1) Memória Compartilhada (*shared memory*) → Neste tipo de arquitetura, todos os processadores podem acessar toda a memória através de um endereço global, ou seja, apesar dos processadores envolvidos operarem independentemente entre si, eles compartilham toda a memória. Isso também implica que qualquer alteração feita na memória por um processador é visível para todos. O que pode diferir é o tipo de acesso à memória, que, para esta classe, pode ser feita de duas formas [127] e [125]:

a. Memória de acesso uniforme (UMA – *Uniform Memory Access*) → Tipo de memória mais comumente utilizado em sistemas com multiprocessamento simétrico (SMP). Os acessos e os tempos de acesso à memória são uniformes para todos os processadores. Esta arquitetura pode ser chamada de memória de acesso uniforme com coerência de cache (CC-UMA - *cache coherent* UMA), quando a alteração de uma posição de memória feita por um processador é imediatamente atualizada para todos os processadores. Na Figura 16 está apresentado um diagrama deste tipo de memória.

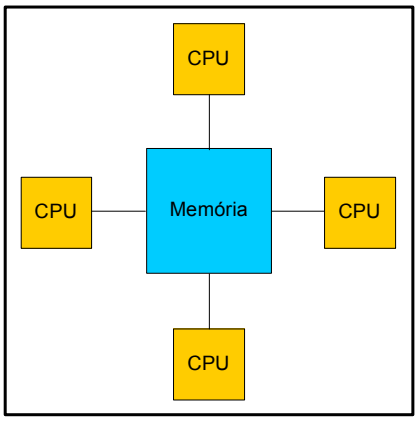

Figura 16 – Memória Compartilhada de Acesso Uniforme

b. Memória de acesso não uniforme (NUMA – *Non-Uniform Memory Access*) → Este tipo de arquitetura de memória é utilizada normalmente em computadores com vários SMP. Toda a memória é compartilhada para todos os processadores, porém o tempo de acesso não é o mesmo, visto que o acesso à memória via barramento de interconexão é mais lento. Tal qual à memória de acesso uniforme, se houver coerência de cache, pode ser chamada de (CC-NUMA -

cache coherent NUMA). O diagrama desta arquitetura está mostrado na Figura 17.

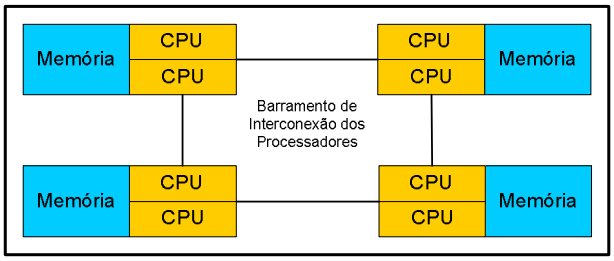

Figura 17 – Memória Compartilhada de Acesso Não Uniforme

As vantagens da memória compartilhada são: endereço global de memória provê uma programação amigável com relação ao uso da memória; o compartilhamento de dados entre as tarefas é rápido por conta da proximidade da memória da CPU. Já as desvantagens são as seguintes: falta de escalabilidade entre memória e CPU, uma vez que o aumento aritmético da quantidade de CPU acarreta um aumento geométrico no tráfego de informação na memória, podendo causar sobrecarga de comunicação (*overhead*); é uma arquitetura cara para a quantidade de processadores dos computadores atuais.

(2) Memória Distribuída (*distributed memory*) → Neste tipo de arquitetura é necessário que exista uma comunicação de rede para conectar as memórias de cada processador. Cada processador possui sua própria memória, que não é mapeada por nenhum outro processador, ou seja, não existe o conceito de memória global. Outra conseqüência é que a alteração da memória feita por um processador não causa nenhum efeito nas outras memórias. Se for necessário que um processador acesse a memória de outro, isso deverá ser explicitamente feito através de instruções no código do programa. Um diagrama com esta arquitetura está mostrado na Figura 18.

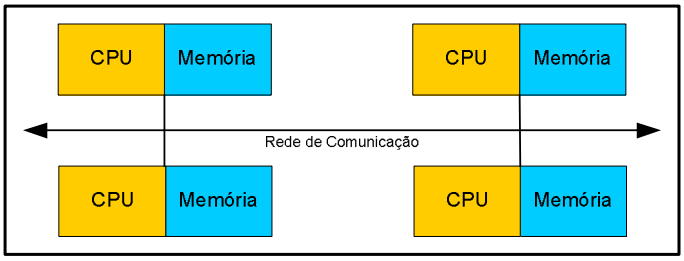

Figura 18 – Memória Distribuída

As vantagens desta arquitetura são as seguintes: ela é escalável, ou seja, ao se conectar mais uma unidade de processamento na rede, a quantidade de memória disponível aumentará; como cada processador possui sua própria memória, não interferindo no acesso dos outros processadores, não existe sobrecarga de comunicação (*overhead*); baixo custo, pois podem ser utilizados computadores comumente encontrados no mercado. As desvantagens são as seguintes: as trocas de dados entre as memórias dos processadores devem ser explicitadas através de instruções específicas; dificuldade de mapear estruturas baseadas em memória global já existentes; não uniformidade de acesso à memória, caso seja necessário acessar a memória de outro processador.

Com a proliferação dos processadores com multiplos núcleos, este tipo de memória está perdendo terreno para o tipo híbrido, descrito a seguir.

(3) Memória Híbrida Distribuída-Compartilhada (*hybrid distributed-shared memory*) → Este tipo de arquitetura é empregado nos mais potentes e rápidos computadores. A memória compartilhada é utilizada dentro da unidade de processamento, geralmente composta de vários processadores (SMP) e a memória distribuída é utilizada por conta da conexão via rede entre várias unidades de processamento. O diagrama desta arquitetura está mostrado na Figura 19. As vantagens e desvantagens desta arquitetura são as mesmas das duas arquiteturas anteriores.

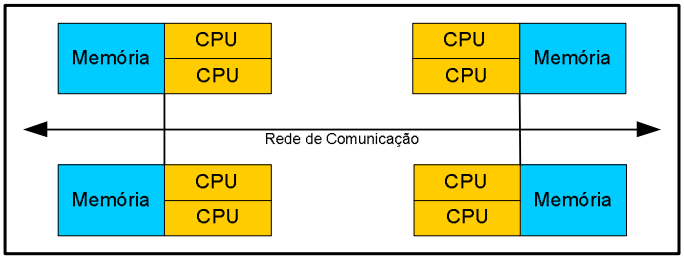

Figura 19 – Memória Híbrida Distribuída-Compartilhada

3.1.2 Cluster

Para entender o que vem a ser um *cluster*, podemos começar com o significado desta palavra. Ao acessar a rede mundial de computadores (*internet*), no sítio da

Universidade de Cambridge está disponível o *Cambridge Advanced Learner´s Dictionary*, onde se pode verificar que o significado da palavra *cluster* é o seguinte: "*a group of similar things that are close together*", ou seja, *cluster* significa um conjunto de coisas similares que estão juntas. Extrapolando este significado para os *clusters* de computadores, poderíamos defini-los como sendo um conjunto de processadores, operando de forma coordenada, com o objetivo de executar uma determinada tarefa.

O desenvolvimento de máquinas com este tipo de arquitetura se deu por conta de quatro razões: barateamento dos processadores de alto desempenho; desenvolvimento de redes de alta velocidade; padronização das ferramentas para desenvolvimento de aplicações paralelas; e do contínuo crescimento maior da demanda por mais processamento do que o oferecido pelos grandes fabricantes de equipamentos.

A idéia de conectar computadores com o intuito de aumentar o poder de processamento não é nova. Para se ter um exemplo, nos anos 70 a IBM possuía um sistema chamado de *Job Entry System* (JES) que permitia facilmente a distribuição de tarefas num *cluster* de grandes computadores proprietários, chamados de *mainframe*.

Em 1994, Thomas Sterling e Don Becker, no centro aeroespacial Goddard da NASA, construíram um computador paralelo utilizando computadores pessoais comuns e programas não proprietários. Este sistema, batizado de Beowulf [1], em homenagem a um herói de um antigo poema épico, era composto de 16 processadores Intel DX4 interligados por uma rede de 10Mbits/s. O sistema operacional utilizado foi o Linux, utilizava compiladores GNU e suportava programação paralela através de mensagens com a utilização da biblioteca MPI, todos estes aplicativos disponíveis gratuitamente. Diante dos resultados, a comunidade de pesquisa com computação de alto desempenho rapidamente abraçou a filosofia de construção utilizada nesta arquitetura. Hoje em dia os grandes fabricantes de computadores estão cada vez mais adotando esta filosofia na montagem das suas máquinas.

As principais características de um *cluster* são as seguintes:

- Desempenho → Atualmente os fabricantes disponibilizam seus principais processadores em computadores pessoais, facilitando a atualização de *clusters* com o estado da arte em termos de processamento;
- Disponibilidade → Caso uma das máquinas apresente defeito, ela pode ser, na pior das hipóteses, retirada do processamento, com a redistribuição das tarefas entre as

que ficaram. Isso é possível porque cada uma das máquinas do *cluster* é um computador completo;

- Baixa relação custo/desempenho → Como se podem utilizar computadores largamente disponíveis no mercado, o custo acaba sendo menor do que a utilização de grandes máquinas proprietárias;
- Crescimento incremental → É possível adicionar mais uma ou quantas máquinas forem necessárias para aumentar o poder de processamento até o ponto desejado (escalabilidade incremental);
- Escalabilidade → Com o aumento de máquinas, todos os parâmetros do cluster, tais como capacidade de processamento, memória e discos para armazenamento, aumentam de forma proporcional. Porém, deve-se atentar para o fato de que, dependendo da aplicação paralela, a rede de comunicação entre as máquinas pode não suportar um grande aumento no tráfego de mensagens;
- Configuração flexível → Caso seja necessária uma conexão de rede mais rápida, de custo mais elevado, pode-se montar o cluster com esta nova rede, mantendo o orçamento final com a diminuição do número de máquinas;
- Compatibilidade e portabilidade → Com a utilização de sistemas padronizados, tais como sistema operacional Linux e sistemas de troca de mensagens (MPI), por exemplo, uma aplicação poderá ser compatível e portável para outros *clusters*.

Por conta destas razões, pode-se perceber o porquê da grande expansão deste tipo de máquina, que hoje em dia é disponibilizado por todos os grandes fabricantes da indústria.

Os componentes básicos de um *cluster* são os seguintes [60]:

- Vários computadores de alto desempenho, que podem ser computadores pessoais, estações de trabalho ou SMP;
- Rede de alta velocidade para conexão entre os computadores;
- *Cluster middleware*, que é uma interface entre as aplicações do usuário com o sistema operacional e os componentes do computador. Esta camada permite que exista o conceito de máquina única, ou seja, o usuário só pode entrar na mesma através de um único ponto, não sendo permitido o acesso aos processadores internos diretamente, conforme pode ser visto na Figura 20. Para o usuário é como se fosse

um microcomputador convencional e os programas básicos é que farão, a partir da quantidade desejada, o gerenciamento de quais processadores será dedicado à aplicação do usuário;

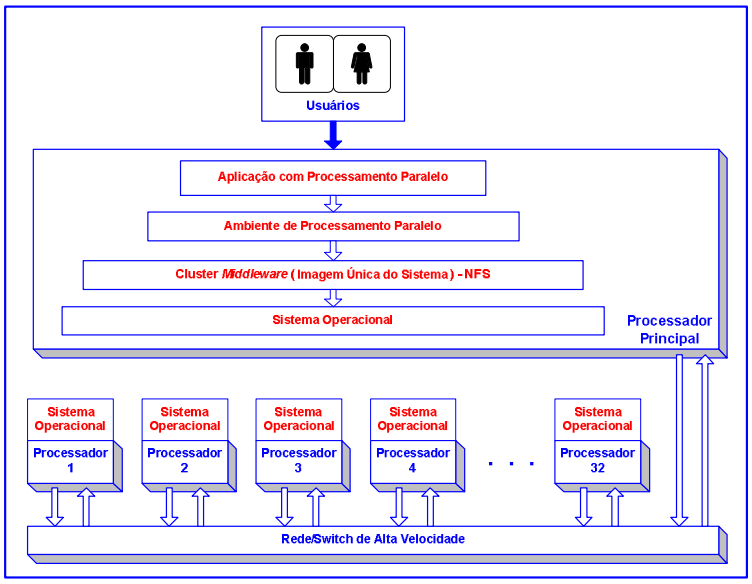

Figura 20 – Conceito de Máquina Única

- Ambiente de programação paralela com disponibilidades de compiladores e dos programas responsáveis pelo gerenciamento das comunicações entre os processadores, tais como MPI, OpenMP e PVM;
- Aplicações dos usuários sejam elas seqüenciais ou paralelas.

Os clusters podem ser classificados de diversas formas, dependendo de quais fatores são levados em consideração. Algumas destas classificações serão apresentadas a seguir [60].

1. Tipo de aplicação:
    a. Alto desempenho → o foco principal é no processamento, normalmente utilizado na área científica;
    b. Alta disponibilidade → o foco principal é na redundância dos equipamentos, sendo utilizado em aplicações críticas onde não pode acontecer parada de processamento.

2. Propriedades dos nós:

    a. Dedicados → todo o cluster executa apenas uma determinada aplicação;
    b. Não dedicados → diversas aplicações pode ser executadas ao mesmo tempo.

3. Tipos de computadores utilizados:

    a. *Cluster* de computadores pessoais;
    b. *Cluster* de estações de trabalho;
    c. *Cluster* de SMP.

4. Configurações dos nós:

    a. Homogêneos → quando todos os nós são constituídos por máquinas iguais, mesmos processadores e mesmas quantidades de memórias, executando os mesmos sistemas operacionais;
    b. Heterogêneos → quando os nós são constituídos por diferentes máquinas e/ou diferentes sistemas operacionais.

5. Sistema Operacional:

    a. Linux → Sistema operacional bastante difundido atualmente, como se pode observar pelo sítio dos 500 maiores computadores do mundo. Possui diversas distribuições, sendo as mais famosas o Red Hat, SUSE, CentOS, Ubuntu, Mandriva etc.;
    b. Solaris → Sistema operacional desenvolvido pela Sun Microsystems. Este sistema possui uma versão de código livre chamado de OpenSolaris;
    c. Windows → Estão disponíveis o Windows Compute Cluster Server 2003 e o Windows HPC 2008;
    d. AIX → Sistema operacional desenvolvido pela IBM, baseado no Unix.

3.1.3 Estatística "*Top 500*"

A proliferação da computação paralela fez com que surgissem muitos sistemas de computadores para processamento paralelo. Para fazer uma classificação destes tipos de computadores, foi criado o sítio www.top500.org, onde podem ser consultadas

diversas estatísticas sobre os 500 computadores mais potentes em atividade no mundo. Estas estatísticas estão disponíveis deste 1993, sendo de publicação semestral. Em termos de arquitetura, dentre os 500 maiores computadores, ao longo do período de apuração, os tipos disponíveis são os seguintes [2]:

- MPP (Abreviação de *Massively Parallel Processors*) – é um grande sistema de processamento paralelo, normalmente composto de milhares de processadores;
- *Cluster* – Conjunto de computadores conectados entre si por uma rede e processando de forma coordenada;
- SMP (*Symmetric Multiprocessors*) – são computadores cujo processador é composto de mais de um núcleo, normalmente são encontrados com dois (*dual core*), quatro núcleos (*quad core*) ou mais núcleos;
- *Constellations* – Também chamadas de *cluster of cluster*, é um *cluster* composto de um conjunto de computadores do tipo SMP com grande quantidade de núcleos de processamento;
- Processador Único (*Single Processor*) – Máquinas que possuem um único processador. Nesta classe ainda pode-se classificar alguns computadores pessoais de custo mais baixo;
- Outros – Outros sistemas de processamento. Pesquisando os dados de 1993, observou-se que estes sistemas são do tipo SIMD, que serão definidos mais adiante.

Uma observação importante sobre a classificação dos 500 maiores computadores do mundo é o crescimento dos sistemas baseados em *cluster*, que surgiu no final da década de 90 e, hoje em dia, domina amplamente a relação destes computadores, conforme pode ser observado na Figura 21 [2], onde quase 90% dos maiores computadores pertencem a essa classe.

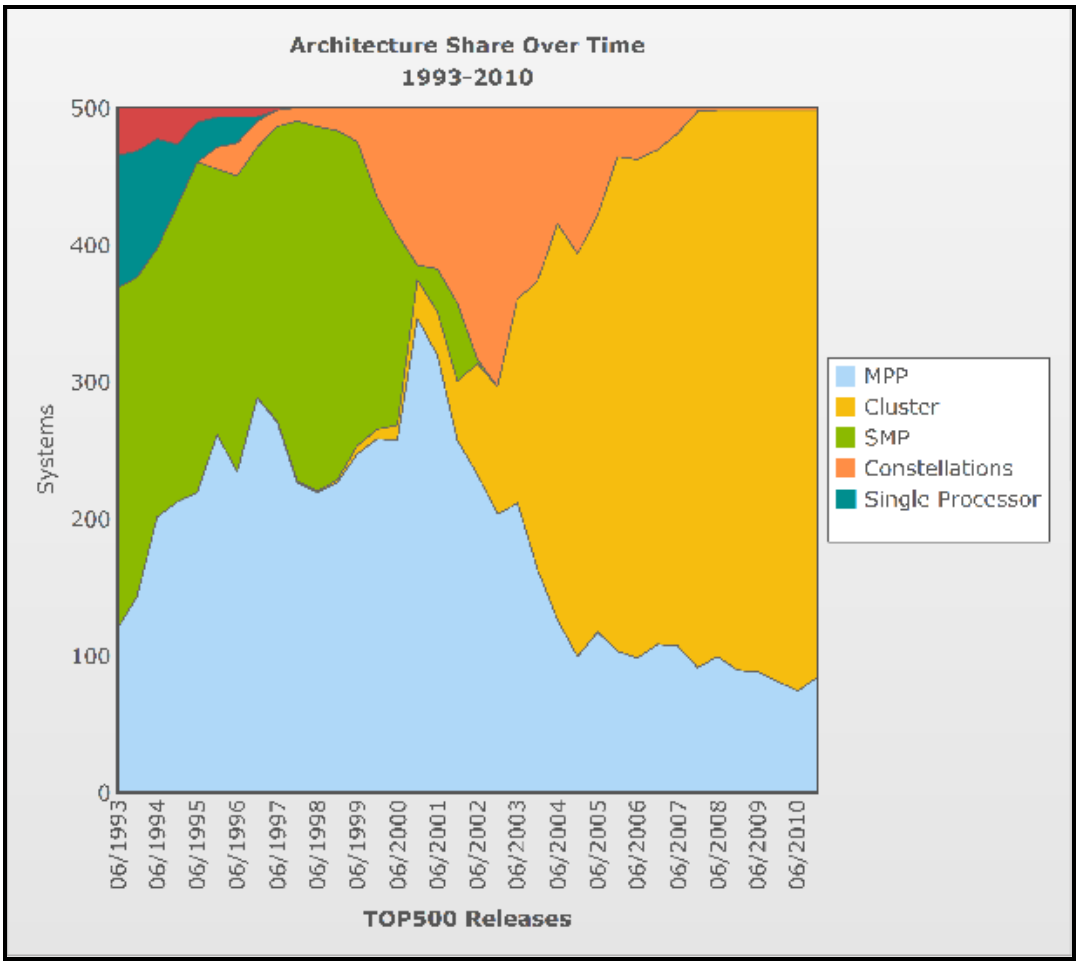

Figura 21 – Histórico das Arquiteturas dos 500 Maiores Computadores

3.2 Modelos de Programação Paralela (*software*)

Em 1988, McGaw e Axelrod, apontaram quatro caminhos diferentes para o desenvolvimento de aplicações para computadores paralelos [1].

(1) Estender um compilador existente para converter programas seqüenciais em programas paralelos.

O desenvolvimento de compiladores para paralelização tem sido uma área bastante ativa de pesquisa por mais de duas décadas, e vários sistemas experimentais foram desenvolvidos. Várias companhias ofereceram compiladores para converter

os programas escritos em Fortran 77 para linguagem paralela, tanto para ambientes de memória compartilhada quanto para memória distribuída.

Esta abordagem nunca foi unânime, pois mesmo que o algoritmo possuísse algum paralelismo, o desenvolvedor poderia tê-lo escondido, sem intenção, numa implementação que não tinha previsão para ser paralela.

(2) Estender uma linguagem de programação com novas instruções para permitir a paralelização dos programas.

Ampliar uma linguagem de programação é o meio mais fácil, rápido, barato e, talvez por estas razões, a mais popular abordagem para a programação paralela. A razão para estas facilidades se deve ao fato de que esta abordagem cria a necessidade apenas da inclusão de uma biblioteca com as novas funções e o compilador utilizado anteriormente continua sendo utilizado. Um exemplo desta abordagem é o MPI, que permite uma grande flexibilidade de implementações de processamento paralelo, de forma rápida, com a inclusão de chamadas para funções especificas.

O lado negativo é que, como o compilador não foi envolvido na geração do código paralelo, ele não identifica erros e existe uma grande dificuldade de se depurar os programas paralelos gerados desta forma.

(3) Criar uma nova camada paralela sobre a linguagem seqüencial existente.

Pode-se imaginar que um programa paralelo tenha duas camadas: uma contendo o núcleo do processamento, na qual os dados são manipulados e os resultados obtidos; a segunda camada controlaria a criação e sincronização dos processos e o particionamento dos dados para os processos. Estas ações poderiam ser programadas utilizando uma linguagem paralela e um compilador seria o responsável para converter este programa de duas camadas num código coerente para execução num computador paralelo. Exemplos desta abordagem são o *Computationally Oriented Display Enviroment* (CODE) e o *Heterogeneous Network Computing Enviroment* (Hence).

Esta abordagem requer o entendimento e o uso de um novo sistema de programação paralela, talvez por isso, não tenha recebido muita atenção por parte da comunidade de programação.

(4) Definir totalmente uma nova linguagem e, conseqüentemente, novos compiladores.

Um exemplo desta abordagem é a linguagem de programação occam. Com uma sintaxe notadamente diferente das tradicionais, esta linguagem suporta tanto a execução paralela quanto a execução tradicional, sendo que a comunicação entre os processos é feita de forma automática.

Outra abordagem seria a de inserir construções paralelas numa linguagem já existente, conforme foi feito com o Fortran 90, HPF e C*.

O problema do desenvolvimento de novas linguagens é que existe uma barreira natural das pessoas em aprendê-las em detrimento de uma que já está dominada. Já o problema de criar um novo compilador é que é necessário muito tempo até a ferramenta ficar madura.

Nos itens a seguir estão descritos os principais modelos utilizados na geração de programas em paralelo [125].

3.2.1 Modelo com Utilização de *Threads* (*Threads Model*)

Neste modelo, um determinado processo pode ter multiplos *threads*, executando diferentes pedaços do código de forma paralela. Este modelo está normalmente associado a ambientes de memória compartilhada.

Pela visão do desenvolvedor, a implementação de *threads* está associada a uma biblioteca de rotinas, a ser utilizada na geração do programa paralelo, e também a um conjunto de diretivas de compilação, que deverão ser inseridas explicitamente no código do programa.

Este tipo de implementação não é recente, porém, por falta de uma padronização, cada fabricante criava a sua própria versão para a implementação de *threads*. Estas versões não eram compatíveis entre sí, o que dificultava a portabilidade das aplicações. Atualmente, as duas implementações mais utilizadas são o POSIX e o OpenMP, sendo que este será comentado a seguir.

3.2.2 Modelo com Utilização de Passagem de Mensagem (*Message Passing*)

Neste modelo os processos são executados em cada uma das máquinas participantes do processamento. A troca de informações entre eles sempre é feita através de mensagens, que necessitam ser explicitamente inseridas no código da aplicação. Esta implementação precisa ser do tipo cooperativa, por exemplo, uma instrução de envio tem que possuir uma instrução de recebimento para que a mensagem ocorra.

Na visão do desenvolvedor, a programação neste modelo também consiste de um conjunto de rotinas a ser utilizada na geração do programa paralelo, assim como de instruções a serem inseridas explicitamente no código do programa com o intuito de definir quais as parcelas do código que devem ser executadas por quais processadores.

Após o surgimento do forum MPI, onde se discutiu amplamente uma semântica e uma sintaxe para a padronização de uma linguagem de passagem de mensagens, o padrão MPI se torno de fato o padrão utilizado pela industria, substituindo várias outras implementações proprietárias. Atualmente existem várias implementações do padrão MPI, o que pode fazer com que um determinado programa necessite ser recompilado para ser executado numa outra máquina que não use a implementação original. Atualmente as duas implementações mais utilizadas são a MPICH, desenvolvido pelo Argonne National Laboratory e a LAM/MPI, desenvolvido pela Universidade de Indiana. O padrão MPI será comentado em um item a seguir.

3.2.3 Modelo de Paralelismo de Dados (*Data Parallel*)

Neste modelo todos os processadores executam um mesmo conjunto de tarefas em subconjuntos diferentes de dados. Em arquiteturas de memória compartilhada, todos os processadores têm acesso automático ao seu conjunto de dados, porém, em arquiteturas de memória distribuída cada conjunto de dados precisa ser dividido e enviado para o respectivo processador. Este processo de envio e recebimento de dados é feito automaticamente pelo compilador e é totalmente transparente.

A programação deste modelo pode ser feita através de chamada de rotinas existente num pacote ou por diretivas de compilação. Um exemplo de implementação deste modelo é o *High Performance Fortran* (HPF), que suporta a programação de paralelização de dados.

3.2.4 Modelo Híbrido

Por conta da existência de muitos *clusters* de processadores do tipo SMP, tornou-se interessante aliar modelos de memória compartilhada com modelos de memória distribuída. Neste contexto, procura-se combinar um modelo que utiliza *threads* (POSIX ou OpenMP) com um que utilize passagem de mensagens (MPI).

3.2.5 MPI (Message Passing Interface)

Em 1992 foi criado um fórum para discutir uma padronização para trocas de mensagens. Este fórum foi composto de pesquisadores, programadores, usuários e fabricantes de computadores que definiram a sintaxe, a semântica e o conjunto de rotinas para troca de mensagens em ambientes com memória distribuída. Esse conjunto de rotinas foi agrupado numa biblioteca chamada de MPI. Estas rotinas, ou funções, são implementadas ao longo do código do programa para viabilizar a distribuição de tarefas entre os processadores participantes do ambiente distribuído. As funções da biblioteca MPI estão disponíveis nas linguagens C e Fortran [128], [129], [130] e [122].

Uma vez iniciado, o ambiente MPI reúne os processos em comunicadores, onde cada processador e cada mensagem ganham rótulos. Através destes rótulos e dos comunicadores se pode gerenciar todas as mensagens do ambiente distribuído, logo, todas as funções do MPI obrigatoriamente se referenciam a estes parâmetros. Outro parâmetro importante é o rótulo identificador da mensagem. Com todos estes parâmetros, chamados de envelope, o comunicador identifica o remetente, o destinatário e a mensagem, transferindo os dados corretamente entre os processadores envolvidos.

A comunicação das mensagens pode ser classificada como ponto-a-ponto, quando envolve dois processadores, ou coletiva quando envolve vários processadores. Um exemplo de comunicação ponto-a-ponto é o par de funções para envio e recebimento de mensagens, *MPI_SEND* e *MPI_RECV*, enquanto que um exemplo de comunicação coletiva é a função *MPI_BCAST*, que transmite os dados de um processador para todos os outros.

A comunicação também pode ser classificada como bloqueante ou não bloqueante. No tipo de comunicação bloqueante, o processador que envia uma mensagem fica esperando até que a memória utilizada para este envio esteja liberada

para uso pelo sistema, o que pode ocorrer antes do início efetivo da transmissão da mesma. O processador que recebe a mensagem fica esperando até que a mesma esteja disponível para uso na sua memória. Já na comunicação não bloqueante esta sincronização não existe e o seu uso melhora o desempenho dos sistemas, desde que estes estejam aptos a suportar uma grande quantidade de mensagens pendentes.

Com relação ao envio de mensagens, ele pode ser feito das seguintes formas, independente se a função é bloqueante ou não:

- Envio síncrono (*synchronous send*) – nesta modalidade o remetente envia uma solicitação de envio de mensagem, que é armazenada pelo destinatário. Quando a instrução de recebimento é iniciada, o destinatário envia a permissão de envio e o remetente envia a mensagem. A instrução de envio pode ser iniciada sem o correspondente início do recebimento, porém só é completada quando o recebimento é finalizado;
- Envio pronto (*ready send*) – nesta modalidade é assumido que o destinatário já está pronto para receber a mensagem e o remetente a envia assim que é processada. Caso o destinatário não esteja efetivamente pronto, o comportamento do programa é indefinido;
- Envio através de memória temporária (*buffered send*) – nesta modalidade ocorre o completo desacoplamento entre a instrução de envio e a de recebimento. Caso o processador destinatário não esteja pronto para receber a mensagem, o processador remetente a copia para uma área de memória temporária (*buffer*). A mensagem só será efetivamente enviada quando a instrução de recebimento for processada no destinatário. Nesta modalidade, deve-se reservar o tamanho da área de memória a ser utilizada na transmissão dos dados e se esse tamanho for insuficiente, ocorrerá um erro no programa;
- Envio padrão (*standard send*) – O ambiente MPI decide como enviar a mensagem. Neste caso, o envio da mensagem pode ser desacoplado do recebimento, já que o MPI pode realizar a comunicação via *buffer*. Porém, caso o espaço de memória reservada utilizado não seja suficiente ou por questões de desempenho, o envio da mensagem passa a depender do processamento da instrução de recebimento. Ou seja, o sistema MPI decide se envia a mensagem na forma síncrona (*synchronous*) ou através de memória temporária (*buffered*).

A biblioteca MPI possui mais de uma centena de funções que permitem uma grande flexibilidade na programação das instruções, porém, com a utilização de um pequeno conjunto de funções é possível fazer com que um programa execute de forma distribuída. Em [129] e em [131] estão apresentadas as descrições das funções disponíveis no MPI.

Outro ponto a destacar é que o MPI não consiste apenas no conjunto de funções reunida na biblioteca MPI disponível em FORTRAN ou C. Para a execução de uma aplicação com processamento distribuído, o MPI disponibiliza um módulo gerenciador de processos, que precisa ser executado em todos os processadores participantes do ambiente paralelo. Este módulo é o responsável em gerenciar as trocas de mensagens, especificadas em instruções dentro da aplicação, utilizando de maneira eficiente os recursos de *hardware* disponíveis para a transmissão de dados entre os processadores.

3.2.6 OpenMP (Open Multi-Processing)

O OpenMP [132], [1], [133] e [134] é uma interface de programação (API em inglês), que permite o desenvolvimento de aplicações com processamento paralelo através da utilização de paralelismo explícito, já que o usuário deve especificar os pontos em que os cálculos deverão ser feitos de forma paralela. Está disponível para plataformas de memória compartilhada, podendo ser utilizada com as linguagens Fortran ou C/C++ e os sistemas operacionais Unix, Linux e Windows. Possui um conjunto de rotinas disponibilizadas numa biblioteca e a sua utilização se dá através da utilização de diretivas de compilação e variáveis de ambiente.

Sua utilização é bastante simples, conforme está mostrado na Figura 22. Nesta figura está apresentado um programa para cálculo do valor de π. O processamento segue apenas no processador principal até que a instrução "`!$OMP PARALLEL DO`" é executada. Esta diretiva informa para o compilador gerar instruções paralelas para a execução do laço da variável "i" que vem a seguir. Junto com a diretiva "`!$OMP PARALLEL DO`" também é disponibilizada a informação de compartilhamento da variável "`w`", através da instrução "`SHARED(w)`", para todos os processadores, assim como o não compartilhamento da variável "`x`", através da instrução "`PRIVATE(x)`". Uma segunda diretiva é a instrução

"`!$OMP& REDUCTION(+: sum)`", que instrui o compilador a gerar instruções para que os resultados parciais da variável "`sum`" em cada processador sejam todos somados no processador principal.

```
      program compute_pi
      integer n, i
      double precision w, x, sum, pi, f, a
c function to integrate
      f(a) = 4.d0 / (1.d0 + a*a)
      print *, 'Enter number of intervals: '
      read *,n
c calculate the interval size
      w = 1.0d0/n
      sum  = 0.0d0
!$OMP PARALLEL DO PRIVATE(x), SHARED(w)
!$OMP& REDUCTION(+: sum)
      do i = 1, n
        x = w * (i - 0.5d0)
        sum = sum + f(x)
      enddo
      pi = w * sum
      print *, 'computed pi = ', pi
      stop
      end
```

Figura 22 – Exemplo de Utilização do OpenMP

Com as diretivas inseridas no código do programa, o OpenMP fará com que as instruções dentro do laço paralelizado sejam distribuídas pelos processadores que fazem parte do ambiente de processamento paralelo disponível.

3.2.7 GPU (Graphics Process Unit)

Uma das tarefas que demandam grande capacidade de processamento é a renderização de gráficos, utilizados principalmente em jogos cada vez mais refinados. As placas gráficas foram criadas originalmente com o objetivo de processar os gráficos dos jogos dos videogames, porém a grande capacidade de processamento de ponto flutuante, aliada ao processamento paralelo necessário para executar as tarefas em tempo real, fizeram com que fosse desenvolvida a possibilidade de se programar algoritmos genéricos, e não só os específicos para processamento gráfico, nos processadores existentes dentro destes aceleradores gráficos. Com a capacidade de se programar genericamente, o termo original GPU, que se refere à unidade de

processamento dedicada ao ambiente gráfico, também passou a ser conhecido como GPGPU, *general purpose computation on graphics processing unit.*

A principal arquitetura de programação utilizada em GPGPU é chamada de CUDA, desenvolvida pela NVIDIA, tradicional fabricante de placas gráficas. Esta arquitetura permite que uma GPU consiga executar as tarefas de processamento gráfico tradicional, assim como tarefas de propósito geral. A linguagem mais utilizada é a CUDA C/C++, que consiste essencialmente na linguagem C/C++ com um conjunto de extensões para permitir a programação nos processadores massivamente paralelos [135]. Convém ressaltar que, desde 2010, está disponível um compilador CUDA para a linguagem de programação Fortran.

Uma grande desvantagem para a utilização de GPU atualmente é a falta de padronização, obrigando a recompilação dos programas caso seja desejado a sua utilização em diferentes placas gráficas.

3.2.8 PVM (Parallel Virtual Machine)

Foi o primeiro programa largamente utilizado para troca de mensagens entre processadores, sendo desenvolvido pelo *Oak Ridge National Laboratories*. A utilização do PVM é bastante similar à do MPI, possuindo um conjunto de rotinas disponibilizadas numa biblioteca para uso com as linguagens Fortran e C e também um programa para gerenciamento das mensagens. É disponibilizado gratuitamente, fato que ajudou muito na sua propagação.

Existem algumas características diferentes entre o PVM e o MPI, porém o que caracteriza bem o MPI é ter um caráter de padronização, o que auxilia na maior difusão deste atualmente.

3.3 Implementação de Programas com Processamento Paralelo (Algoritmos)

No desenvolvimento de programas ou algoritmos em processamento paralelo, alguns cuidados extras devem ser tomados para melhorar o desempenho final da aplicação. Alguns destes cuidados estão descritos nos itens a seguir [121], [1], [122], [123], [124], [125], [136], [137], [138] e [139].

3.3.1 Conhecer o Problema e o Programa

Apesar de ser algo básico, convém ressaltar a importância que o desenvolvedor conheça, de preferência profundamente, tanto o problema quanto o programa. Desta forma o programa paralelo toma forma muito mais rápido e eventuais correções do código também são mais facilmente executadas.

Normalmente o que acontece na prática é a proposta de se paralelizar um programa que é executado serialmente. Neste caso, conhecer o programa ajuda na identificação e eliminação de gargalos de programação, porém, o conhecimento do problema pode ajudar na mudança do algoritmo de solução para a execução paralela, diminuindo significativamente o tempo de execução do programa.

3.3.2 Particionamento do Problema

Existem basicamente duas formas de se particionar um problema: decomposição de domínio e decomposição funcional.

Na decomposição de domínio, os dados associados ao problema são divididos para cada processo paralelo.

Na decomposição funcional, o foco é o processamento e não a manipulação dos dados. Neste caso, o problema é decomposto de acordo com o trabalho a ser feito, logo, cada processo executa uma parcela do trabalho. Exemplos deste tipo de particionamento são modelagem de ecosistemas, processamento de sinais e modelagem climática etc.

3.3.3 Tipos de Comunicação

Na hora de decidir sobre o uso de comunicação entre processadores, deve-se ter em mente os parâmetros descritos a seguir.

- Comunicações síncronas e assíncronas

As comunicações síncronas são chamadas de comunicações bloqueantes, uma vez que elas bloqueiam o processamento até que a comunicação seja finalizada. Neste ponto, chamado de ponto de sincronismo, pode significar um bom tempo ocioso, uma vez que um determinado processador pode ficar bastante tempo parado a espera do fim da comunicação por conta de algum atraso no processamento de outro processador.

As comunicações assíncronas são chamadas de não bloqueantes porque o processamento continua enquanto a comunicação é feita e essa é a grande vantagem deste tipo de comunicação.

- Escopo de comunicação

A comunicação pode ocorrer apenas entre dois processadores, neste caso é chamada de ponto-a-ponto. Neste caso deve-se especificar uma instrução de envio de dados num processador e outra de recebimento de dados em outro processador. Esta comunicação pode ser síncrona ou assíncrona.

Quando a comunicação envolve todos os processadores de um grupo, ela é chamada de coletiva. Neste caso, uma mesma instrução é executada por todos os processadores, sempre de forma bloqueante.

- Parâmetros de comunicação

Os dois parâmetros de suma importância numa rede de comunicação são a latência (*latency*) e a largura de banda (*bandwidth*) [127]. A latência de uma rede é o tempo que se leva para enviar uma mensagem mínima de um ponto A para um ponto B. É normalmente expressa em microsegundos. A largura de banda é a quantidade de dados que se pode comunicar por unidade de tempo. É normalmente expressa em megabytes/segundo ou gigabytes/segundo.

Normalmente é muito melhor enviar uma grande mensagem, para aproveitar bem a largura de banda, do que enviar diversas mensagens pequenas, que poderão ocasionar congestionamento por causa da latência da rede.

3.3.4 Balanceamento de Carga

Este termo se refere à distribuição de trabalho para os processadores, de forma a mantê-los trabalhando o tempo todo. Ou seja, o objetivo do balanceamento de carga é minimizar o tempo ocioso dos processadores, pois quanto menor este tempo, melhor será o desempenho do programa.

Uma forma comum de balanceamento de carga, e fácil de se implementar, é enviar quantidades iguais de problemas para os processadores. Porém, com este balanceamento estático é muito difícil se conseguir um bom balanceamento porque, em

geral, uma quantidade igual de problemas não significa uma quantidade igual de processamento. Muitos problemas "parecidos" levam tempos muito diferentes para serem resolvidos.

Normalmente um balanceamento de carga dinâmico fornece resultados muito melhores. Neste caso, um processo fica responsável de controlar os processadores que terminam seus conjuntos de problemas, de forma a enviar outros conjuntos de problemas e manter todos os processadores em operação. O problema é que este tipo de implementação não é tão simples quanto o anterior.

3.3.5 Conceito de Granularidade

Em computação paralela, a granularidade é uma medida qualitativa dos tempos gastos com o processamento e a comunicação, uma vez que períodos de processamento são normalmente separados por períodos de comunicação, através de eventos de sincronização. Partindo-se desta definição, são adotados dois tipos de granularidade: a fina e a grossa.

A granularidade fina é aquela em que os períodos de processamento entre eventos de comunicação são curtos, resultando numa relação processamento-comunicação baixa. Neste tipo de granularidade podem ocorrer problemas de comunicação por conta de sobrecarga no tráfego de informações. Outra característica importante é que a busca por melhoria de desempenho do algoritmo fica bastante prejudicada. Uma vantagem desta granularidade é que, normalmente, o balanceamento de carga é mais fácil de se obter.

O oposto da granularidade fina é a chamada granularidade grossa, onde os períodos de processamento são grandes, resultando numa relação processamento-comunicação alta. Caem nesta classificação os problemas chamados embaraçosamente paralelos. Nesta classe existem muito mais oportunidades de melhoria do desempenho do algoritmo, porém o balanceamento de carga é mais difícil de ser conseguido.

3.3.6 Instruções de Entrada/Saída (I/O)

Operações de escrita e leitura são inibidoras de paralelismo. Múltiplas instruções de escrita num mesmo arquivo, por exemplo, pode resultar em sobreescrita de dados.

No caso de múltiplas instruções de leitura, pode haver demora na execução da instrução por conta do sistema operacional ter que tratar várias requisições de leitura ao mesmo tempo.

3.3.7 Medidas de Desempenho

Um parâmetro importante numa aplicação com processamento paralelo é a medida do seu desempenho. Normalmente podem ser utilizadas duas medidas de desempenho, que estão descritas a seguir.

(1) Fator de aceleração (ou *speedup*) [121]:

$$S_p = \frac{t_1}{t_p} \tag{45}$$

Esta medida é obtida pela divisão do tempo real de execução da aplicação com um processador (execução convencional) pelo tempo de execução com "p" processadores.

(2) Eficiência:

$$\eta_p = \frac{S_p}{p} \times 100\% = \frac{t_1}{p \cdot t_p} \times 100\% \tag{46}$$

Esta medida é obtida pela divisão do fator de aceleração pela quantidade de processadores utilizados na execução em paralelo.

Estas medidas funcionam da seguinte forma: imagine uma aplicação que leve um determinado tempo para terminar uma execução e fornecer os resultados. É natural achar que, se a mesma for paralelizada e executada com dois processadores, ela leve metade do tempo que levava na forma convencional de execução. Para este caso, o fator de aceleração será dois e a eficiência será de 100%. Extrapolando este raciocínio, chega-se à conclusão que o fator de aceleração ideal será sempre igual ao número de processadores utilizado no processamento paralelo, o que significa uma eficiência de 100%. Traçando um gráfico do fator de aceleração em função do número de

processadores, o ideal é encontrar uma reta de quarenta e cinco graus, o que é comumente chamado de *speedup* linear, conforme pode ser visto na Figura 23.

Normalmente o gráfico do fator de aceleração em função do número de processadores é uma curva que satura, ou seja, o crescimento da quantidade de processadores é maior do que o crescimento do fator de aceleração. Com isso a curva fica situada abaixo da reta do *speedup* linear, significando que as eficiências estão abaixo de 100%. Eventualmente existem casos em que curva do fator de aceleração fica acima da reta de *speedup* linear, indicando eficiências maiores de 100%, o que implica na ocorrência do chamado *speedup* superlinear, que é a região destacada na Figura 23. Caso este fato aconteça na comparação com o algoritmo serial, na execução monoprocessada, isso pode ter como causa a utilização de um algoritmo mais eficiente na execução paralela do que na convencional. Esta dúvida pode ser desfeita com a utilização da execução paralela com um processador em vez da execução convencional, já que neste caso os algoritmos são os mesmos. O surgimento de *speedup* superlinear nesta comparação está associado com características da arquitetura do processador que está sendo utilizado. Conforme a quantidade de processadores aumenta, o problema pode ser tornar pequeno suficiente para que caiba totalmente na memória cache do processador. Como o acesso a este tipo de memória é muito mais rápido que o acesso à memória RAM convencional, o programa pode se tornar muito mais rápido na execução paralela, gerando eficiências maiores de 100%.

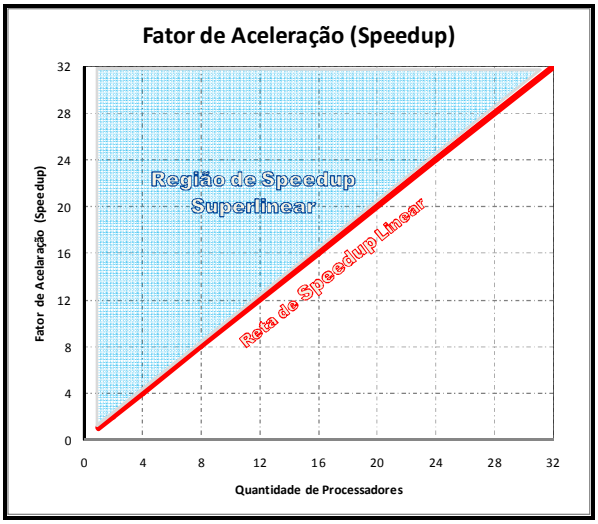

Figura 23 – Fator de Aceleração ou *Speedup*

As medidas do fator de aceleração e da eficiência fornecem boas avaliações quando todo o código da aplicação pode ser paralelizado, porém isso é quase impossível de acontecer na prática. Se a parcela não paralelizável da aplicação computacional for significativa, os valores obtidos com estas medidas poderão não ser animadores. Vamos supor que uma determinada aplicação será paralelizada, porém o percentual a ser paralelizado não é de 100% e sim uma das três situações que estão mostradas na Figura 24 [60]. Nestes três exemplos a parcela paralelizável é de 90%, 50% e 10% do tempo total da aplicação, respectivamente.

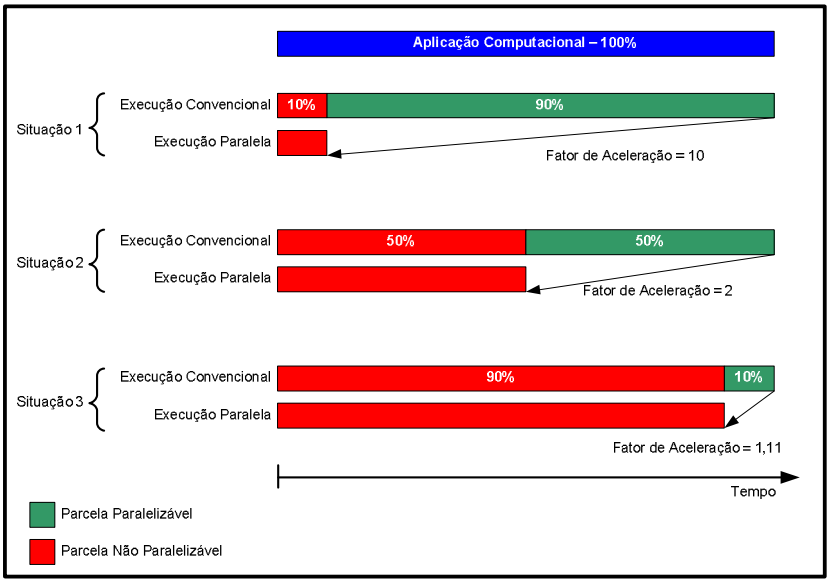

Figura 24 – Comparação de Fatores de Aceleração

Outra suposição seria que, ao executar a aplicação de forma paralela, utilizássemos uma grande quantidade de processadores tal que o tempo gasto na parte paralela ficasse nulo. Evidentemente que isto é uma abstração da realidade, uma vez que a quantidade de processadores necessária para a ocorrência deste fato seria infinita e, além disto, não deveria existir nenhum tempo utilizado para comunicação e o balanceamento de carga deveria ser perfeito. Pode-se perceber que os fatores de aceleração destes três casos seriam relativamente pequenos a as suas eficiências seriam baixas, uma vez que a quantidade de processadores necessária para desprezar a parte paralela não seria pequena. O tempo mínimo de execução destes três exemplos seriam, respectivamente, 10%, 50% e 90% do tempo da aplicação com um processador,

resultando em fatores de aceleração de 10 (100%/10%), de 2 (100%/50%) e de 1,11 (100%/90%), respectivamente.

Será que a paralelização destes programas valeria à pena? Somente a análise de cada situação é que poderá responder a essa pergunta. Observando a situação em que uma aplicação possui 50% do seu código paralelizável. Se o tempo da execução serial for de trinta minutos e a aplicação seja executada uma vez por mês, certamente o esforço que será despedido na paralelização do seu código fonte, para ganhar 15 minutos, não valerá à pena. Agora, se a aplicação levar 30 dias na execução convencional, certamente o esforço de paralelizá-la valerá a pena. Logo, situações projetadas de mesmo fator de aceleração podem ter decisões diametralmente opostas de realizá-las ou não.

3.3.7.1 Lei de Amdahl

Como demonstrado no item anterior, na prática todo programa que pode ser paralelizável possui uma parte do código serial e este fato impõe um limite no fator de aceleração. É disto que trata a chamada lei de Amdahl [140], que trata a fórmula do cálculo do fator de aceleração através da divisão do tempo total de execução em duas partes, uma referente à parte serial e outra referente à parte paralelizável, conforme está mostrado na Figura 25 [121].

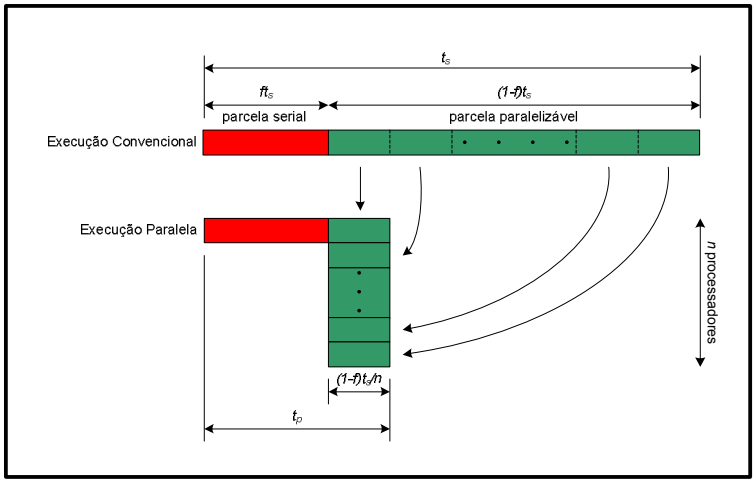

Figura 25 – Lei de Amdahl

Os tempos totais das execuções serial e paralela, com n processadores, podem ser obtidos através das seguintes expressões:

$$tempo\ total\ serial = t_s = parcela\ serial + parcela\ paralela \tag{47}$$

$$tempo\ total\ \ paralelo = t_p = parcela\ serial + \frac{parcela\ paralela}{n} \tag{48}$$

Supondo que a parcela serial do programa leve uma fração "f" do tempo total, evidentemente que a parcela paralela levará o restante do tempo gasto, conforme pode ser observado nas expressões a seguir.

$$parcela\ serial = ft_s, \qquad 0 \le f \le 1 \tag{49}$$

$$parcela\ \ paralela = (1-f)t_s \tag{50}$$

Substituindo (49) e (50) em (48), tem-se.

$$t_p = ft_s + \frac{(1-f)t_s}{n} \tag{51}$$

Substituindo (51) em (45), tem-se.

$$S_p = \frac{t_1}{t_p} = \frac{t_s}{t_p} = \frac{t_s}{ft_s + \frac{(1-f)t_s}{n}} = \frac{t_s}{\frac{nft_s + t_s - ft_s}{n}} = \frac{nt_s}{t_s(nf+1-f)} = \frac{n}{1+(n-1)f} \tag{52}$$

A expressão obtida em (52) permite que se calcule o fator de aceleração máximo possível através da aplicação do conceito de limite.

$$\lim_{n\to\infty} S_p(n) = \frac{1}{f} \tag{53}$$

Pela lei de Amdahl, a fração serial do programa define o valor máximo possível do fator de aceleração da execução paralela deste mesmo programa. Observando os valores dos fatores de aceleração das três situações apresentadas na Figura 24, verifica-se que os mesmos estão de acordo com a expressão (53). Se um programa tiver uma parcela de 10% de processamento serial, o fator de aceleração máximo que se obterá será de 10, ou seja, mesmo que sejam utilizados centenas de processadores, na melhor das hipóteses, o programa levará um décimo do tempo original. Amdahl utilizou este

argumento para promover a execução convencional, em detrimento do paralelismo, no final da década de 1960. Convém novamente lembrar que a decisão de utilizar processamento paralelo depende de cada caso, pois mesmo um fator de aceleração de 10 pode ser muito bom em termos de tempo total economizado.

3.3.7.2 Lei de Gustafson

John L. Gustafson [141], trabalhando numa pesquisa envolvendo processamento massivamente paralelo, encontrou resultados de fator de aceleração bastante diferentes daqueles calculados através da lei de Amdhal. Utilizando um sistema de 1024 processadores, fatores de aceleração de 1016, 1020 e 1021 foram encontrados para algumas das aplicações testadas. Para se ter uma idéia, pela lei de Amdahl, a fração serial deveria ser aproximadamente de 0,000287% para que um sistema de 1024 processadores consiga um fator de aceleração de 1021. Gustafson concluiu que a lei de Amdhal considera que a parte paralela independe da quantidade de processadores, o que normalmente não é verdade. Na prática, em muitos tipos de problemas, o tamanho é escalável com o número de processadores, ou seja, o aumento da quantidade de processadores permite aumentar o tamanho do problema a ser resolvido sem aumentar significativamente o tempo total gasto na parte paralela. Ou seja, o problema é constante em termos de tempo total de execução e não em tamanho, conforme propõe a lei de Amdhal.

Pela lei de Gustafson, o tempo total da execução paralela é composto de uma parcela serial e outra paralela. Já a execução serial, caso fosse possível a execução do mesmo problema da execução paralela em ambiente monoprocessado, deveria ser igual à parcela serial, assumida constante, acrescida de "n" vezes a parcela paralela, resultado da escalabilidade do problema. Estas expressões estão mostradas a seguir.

$$tempo\ total\ paralelo = t_p = parcela\ serial + parcela\ paralela \tag{54}$$

$$tempo\ total\ serial = t_s = parcela\ serial + n \times parcela\ paralela \tag{55}$$

Assumindo que as parcelas serial e paralela possuam as características das equações (49) e (50) e substituindo-as na equação (45), tem-se:

$$S_p = \frac{t_1}{t_p} = \frac{t_s}{t_p} = \frac{ft_s + n(1-f)t_s}{ft_s + (1-f)t_s} = f + n(1-f) = n + (1-n)f \quad (56)$$

A expressão final de (56) é conhecida como lei de Gustafson e o fator de aceleração obtido é conhecido como fator de aceleração escalável (*scaled speedup*).

Para efeito de comparação, estão apresentados na Tabela 1 os valores dos fatores de aceleração, calculados tanto pela lei de Amdhal quanto pela lei de Gustafson, para vários valores de percentuais da parcela serial.

Tabela 1 – Comparação dos Fatores de Aceleração (Lei de Amdhal x Lei de Gustafson)

Percentual da Parcela Serial n = 32 Processadores	Fatores de Aceleração		Relação Gustafson/Amdahl
	Lei de Amdhal	Lei de Gustafson	
0	32.000	32.000	1.000
5	12.549	30.450	2.426
10	7.805	28.900	3.703
20	4.444	25.800	5.805
30	3.107	22.700	7.307
40	2.388	19.600	8.208
50	1.939	16.500	8.508
60	1.633	13.400	8.208
70	1.410	10.300	7.307
80	1.240	7.200	5.805
90	1.107	4.100	3.703
95	1.051	2.550	2.426
100	1.000	1.000	1.000

Uma conclusão que se chega da análise dos resultados mostrados na Tabela 1 é que a relação entre os fatores de aceleração apresenta um máximo quando o percentual da parcela serial é de 50%.

A visão destas duas leis é diametralmente oposta, a lei de Amdhal possui uma visão pessimista, enquanto que a lei de Gustafson possui uma visão otimista do processo de paralelização. O que ocorre na prática é que o fator de aceleração da execução em paralelo estará em algum ponto entre os limites impostos pelas duas leis [60].

3.3.8 Perfilagem

Antes da determinação de uma solução de processamento paralelo para uma aplicação já existente, é necessário realizar alguns, ou vários, ciclos de desempenho completo na versão de execução serial. Este ciclo, que está mostrado na Figura 26, consiste de cinco fases: (1) obter alguma medida de desempenho da aplicação. Este ato

de medir os tempos gastos em cada rotina é chamado de perfilar um programa e a ferramenta utilizada para este fim é chamado de perfilador; (2) análise das informações, procurando identificar possíveis problemas no código do programa; (3) solução para os problemas detectados na fase anterior; (4) implementação das soluções propostas no código do programa; e (5) testar a nova versão do programa, obtendo novas medidas de desempenho e continuar o ciclo até que os resultados possam ser considerados bons.

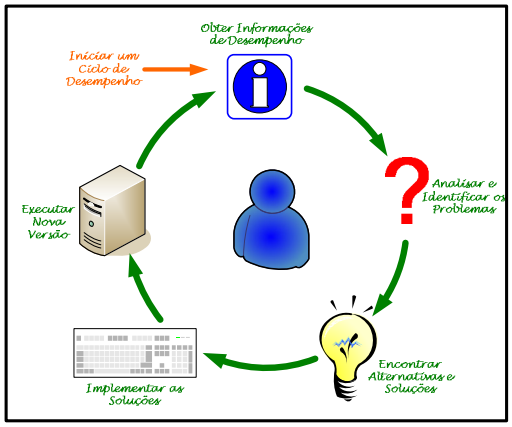

Figura 26 – Ciclo de Desempenho

3.3.9 Outras Características

Outras características que devem ser levadas em conta são as seguintes: portabilidade, requisitos de sistema e escalabilidade.

Graças à padronização dos modelos de computação paralela, em especial o MPI, tornou-se possível que uma aplicação implementada num computador seja executada em outro. Porém, convém verificar se será necessário algum tipo de portabilidade para escolher bem as ferramentas que serão utilizadas no desenvolvimento do programa ou algoritmo.

Com relação aos requisitos de sistema, convém esclarecer que a execução paralela de um programa exige muito mais processamento e memória do que a execução serial, uma vez que muito mais processadores estão trabalhando em conjunto para a solução de um caso.

A escabilidade é a capacidade do programa ou algoritmo manter o desempenho com o aumento dos casos que deverão ser resolvidos. É importante ter em mente que,

com a mudança da forma de executar um programa, da forma serial para a forma paralela, os casos que serão exigidos tendem a ser muito maiores. Logo, é bom desenvolver um algoritmo que não perca muito desempenho com o aumento dos problemas a serem resolvidos.

Capítulo 4.

Estratégia de Paralelização Inicial

Neste capítulo está apresentada a teoria da metodologia proposta para aplicação de processamento paralelo no problema de planejamento da operação de sistemas hidrotérmicos.

4.1 Teoria da Metodologia Paralela Proposta

Com o intuito de facilitar o entendimento da notação utilizada neste trabalho, os conceitos de período, ciclos, série hidrológica, cenários de afluência e iteração estão mostrados na Figura 27.

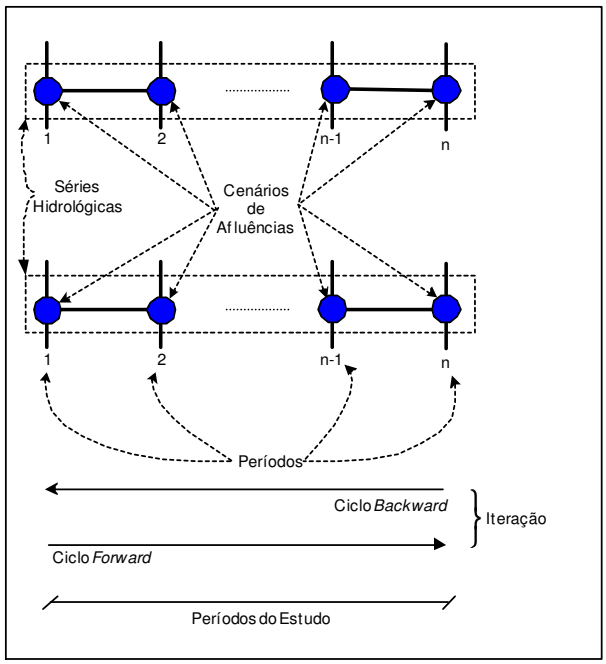

Figura 27 – Definição de Parâmetros Utilizados na Solução do Problema de Planejamento da Operação

Os principais conceitos são os seguintes:

- Período → tem a ver com a discretização do problema, que neste caso é mensal.
- Cenários de Afluências → tem a ver com a série hidrológica que define os valores das afluências de todos os períodos do estudo. Para aumentar a qualidade dos resultados, são utilizadas várias séries hidrológicas na solução do problema.
- Ciclo → está relacionado com a etapa do processo de solução. Na etapa de solução em que os problemas são resolvidos do final para o início, este ciclo é chamado de *backward*, enquanto que a etapa que se inicia no primeiro período é chamada de *forward*.
- Iteração → está relacionada com uma execução completa dos ciclos *backward* e *forward*.

4.1.1 Detalhamento da Solução Seqüencial

A solução do problema de planejamento da operação consiste na determinação, para cada período do estudo, de uma estratégia, composta pelas metas de geração para cada subsistema, de forma a minimizar o valor esperado do custo total de operação. Estas metas são representadas pela FCF, que é obtida de forma aproximada por uma função linear por partes e construída iterativamente pelos cortes de Benders. Para cada estágio e para cada estado do sistema, representado pelos níveis de armazenamento e afluências nos meses anteriores, o problema de operação hidrotérmica é modelado como um problema de programação linear (PL) e as variáveis duais associadas à solução desse problema são utilizadas para a construção dos cortes de Benders.

Como já foi descrito anteriormente, esta solução, quando obtida com a aplicação de PDDE, consiste num processo iterativo para aproximações sucessivas da solução ótima de um problema de programação dinâmica estocástica, onde a estocasticidade do problema está representada pelos vários cenários de afluência para cada reservatório modelado. O processo iterativo possui duas etapas: *forward* e *backward*.

O processo *forward* consiste na solução sucessiva dos problemas do início para o fim do período de estudo. O objetivo desta fase é estimar o valor esperado do custo de

operação levando-se em consideração a FCF construída no processo *backward* imediatamente anterior.

O processo *backward* consiste na solução dos problemas do fim para o início do período de estudo. Nesta fase são incluídas no problema do período anterior, que é resolvido posteriormente, as restrições obtidas no problema do período corrente (cortes de Benders).

Para um determinado período do planejamento, seja no processo *forward* ou no *backward*, serão resolvidos tantos problemas quantos forem os cenários de afluências em cada reservatório de cada subsistema. Estes problemas partem das mesmas condições iniciais, que são os níveis de armazenamento de cada reservatório e, em função da quantidade de energia recebida pelas afluências, calculam as metas de geração das usinas hidráulicas e térmicas para atender à demanda do mercado. A solução do problema de um cenário é totalmente independente da solução dos outros.

O processo iterativo da PDDE pode começar com uma execução do ciclo *forward*, porém, neste caso, a solução do PL somente teria a parcela referente à FCP, uma vez que a FCF é construída na passagem pelo ciclo *backward*. Para esta situação, o problema descrito na seção 2.6, representado pelas equações (18) a (26), é montado da seguinte forma.

$$\alpha_{cen}^{1,F,k} = Min\left\{\sum_{R}\left[C_T\left(GT(R,k)\right) + C_D\left(DEF(R,k)\right)\right]\right\}$$

Sujeito a

$$EA(R,k+1) = EA(R,k) + EC(R,k) - GH(R,k) - EVERT(R,k) - EVAP(R,k) - EVMOR(R,k), \forall R$$

$$GH(R,k) + GT(R,k) + \sum_{s \in R}\left(INT(s,R,k) - INT(R,s,k)\right) = DEM(R,k) - DEF(R,k), \forall R$$

$$\underline{GH}(R,k) \le GH(R,k) \le \overline{GH}(R), \forall R \qquad (57)$$

$$\sum_{\forall s \ne R} INT(s,R,k) - \sum_{\forall s \ne R} INT(R,s,k) = 0, \forall R$$

$$\underline{INT}(s,R) \le INT(s,R,k) \le \overline{INT}(s,R), \forall s \ne R, \forall R$$

$$\underline{GT}(R) \le GT(R,k) \le \overline{GT}(R), \forall R$$

$$\underline{EA}(R,k+1) \le EA(R,k+1) \le \overline{EA}(R,k+1), \forall R$$

Onde:

$k \rightarrow$ Período.

$cen \rightarrow$ Cenário de afluência.

$\alpha_{cen}^{1,F,k} \rightarrow$ Valor esperado do custo de operação do período k do primeiro ciclo *forward*, para o cenário de afluência *cen*.

$R,s \rightarrow$ Subsistemas.

$C_T \rightarrow$ Função que calcula o custo de geração térmica.

$GT \rightarrow$ Geração das usinas térmicas de um subsistema num determinado período. Seu valor máximo é representado por $\overline{GT}$ e seu valor mínimo é representado por $\underline{GT}$.

$C_D \rightarrow$ Função que calcula o custo do não atendimento da demanda.

$DEF \rightarrow$ Demanda não atendida (*deficit*) de um subsistema num determinado período.

$EA \rightarrow$ Energia armazenada de um subsistema num determinado período. Seu valor máximo é representado por $\overline{EA}$ e seu valor mínimo por $\underline{EA}$.

$EC \rightarrow$ Energia controlável de um subsistema num determinado período.

$GH \rightarrow$ Geração das usinas hidráulicas de um subsistema num determinado período. Seu valor máximo é representado por $\overline{GH}$.

$EVERT \rightarrow$ Energia perdida através de vertimento de um subsistema num determinado período.

$EVAP \rightarrow$ Energia perdida através de evaporação de um subsistema num determinado período.

$EVMOR \rightarrow$ Energia perdido por conta do enchimento de volume morto de um subsistema num determinado período.

$INT \rightarrow$ Intercâmbio de energia entre dois subsistemas num determinado período. Seu valor máximo é indicado por $\overline{INT}$.

$DEM \rightarrow$ Demanda de mercado a ser suprida num subsistema num determinado período.

Este seria o problema a ser resolvido em todos os períodos da primeira passagem pelo ciclo *forward*. Pode-se perceber que nesta primeira passagem não existe a parcela referente à FCF nem os cortes de Benders, pois tanto a FCF quanto os cortes de Benders só serão incluídos na formulação do problema na passagem pelo ciclo *backward*. A ausência destes parâmetros faz com que os PLs a serem resolvidos não levem em consideração o custo futuro da decisão tomada no presente. Logo, a solução priorizará a utilização da geração hidráulica para suprir a demanda, provocando maiores custos de operação para períodos futuros por conta da falta de energia armazenada.

Na prática, a solução iniciando-se pelo ciclo *forward* não é comumente realizada porque a utilização maciça de energia hidráulica faz com que os estados de armazenamento utilizados na solução não sejam realistas, pois os reservatórios estarão

vazios durante boa parte do estudo, fazendo com que os custos de operação do sistema sejam muito altos nas primeiras iterações. Para evitar este problema, o processo iterativo inicia-se comumente no ciclo *backward*.

A execução do primeiro ciclo *backward* antes do ciclo *forward*, apresenta um problema, que é a determinação dos estados de armazenamento dos reservatórios em cada período, já que os mesmos são obtidos durante a execução do ciclo *forward*. Logo, uma tarefa importante no início do ciclo *backward* é a inicialização dos estados de armazenamento dos reservatórios, $EA(R,k)$, em todos os períodos do estudo. Em [142], Morton apresenta quatro métodos para realizar a inicialização do primeiro ciclo *backward*.

Depois da determinação dos estados iniciais, o processo de solução da PDDE é descrito da seguinte forma.

- **Primeiro Ciclo da Solução *Backward***

O principal objetivo do ciclo *backward* é determinar os cortes de Benders para cada período de estudo. Estes cortes fornecem as conseqüências futuras das decisões tomadas nos períodos presentes, ou seja, eles compõem a FCF. O primeiro período a ser resolvido é o último período de estudo e a primeira tarefa deste ciclo é determinar os cortes de Benders deste período, que serão levados em consideração no penúltimo período do estudo, que é o segundo a ser resolvido. Cada um dos problemas referentes a um cenário de afluência gera um corte de Benders e todos eles serão representados em forma de restrições para o problema do penúltimo período, que fica na forma apresentada em (58).

$$\alpha_{cen}^{1,B,k-1} = Min\left\{\sum_{R}\left[C_T\left(GT(R,k-1)\right)+C_D\left(DEF(R,k-1)\right)\right]\right\}+\frac{1}{1+\beta}\alpha^{1,B,k}$$

Sujeito a

$$\begin{aligned}
&EA(R,k) = EA(R,k-1)+EC(R,k-1)-GH(R,k-1)-EVERT(R,k-1)-EVAP(R,k-1)-EVMOR(R,k-1),\ \forall R\\
&GH(R,k-1)+GT(R,k-1)+\sum_{s\in R}\left(INT(s,R,k-1)-INT(R,s,k-1)\right) = DEM(R,k-1)-DEF(R,k-1),\ \forall R\\
&\underline{GH}(R)\le GH(R,k-1)\le \overline{GH}(R),\forall R\\
&\sum_{\forall s\neq R} INT(s,R,k-1)-\sum_{\forall s\neq R} INT(R,s,k-1)=0,\ \forall R
\end{aligned} \qquad (58)$$

$$\underline{INT}(s,R) \le INT(s,R,k-1) \le \overline{INT}(s,R), \forall s \ne R, \forall R$$

$$\underline{GT}(R) \le GT(R,k-1) \le \overline{GT}(R), \forall R$$

$$\underline{EA}(R,k) \le EA(R,k) \le \overline{EA}(R,), \forall R$$

$$\alpha_1^{1,B,k} - \sum_R \pi_1^{1,k} EA_1^{1,k}(R,k) \cdot EA(R,k) \ge \delta_1^{1,k}$$

$$\alpha_2^{1,B,k} - \sum_R \pi_2^{1,k} EA_2^{1,k}(R,k) \cdot EA(R,k) \ge \delta_2^{1,k}$$

$$\vdots$$

$$\alpha_c^{1,B,k} - \sum_R \pi_c^{1,k} EA_c^{1,k}(R,k) \cdot EA(R,k) \ge \delta_c^{1,k}$$

Onde:

$\alpha_{cen}^{1,B,k-1} \rightarrow$ Valor esperado do custo de operação do período *k-1* para o cenário de afluência *cen* no primeiro ciclo *backward*.

$\alpha^{1,B,k} \rightarrow$ Variável escalar representando o valor esperado do custo de operação do período *k* no primeiro ciclo *backward*.

$\beta \rightarrow$ Taxa de desconto de um valor num período futuro para o valor presente.

$\pi_{cen}^{1,k} EA_{cen}^{1,k} \rightarrow$ Derivada da função objetivo em relação à energia armazenada de um subsistema no período *k* da primeira iteração, para o cenário *cen*. A quantidade destas restrições é função do número do corte de Benders (cenário de afluência) e do número de iterações.

$\delta_{cen}^{1,k} \rightarrow$ Termo constante da restrição linear do corte de Benders.

$c \rightarrow$ Corte de Benders.

Em cada período de solução, vários PLs serão resolvidos, onde em cada um deles estará representado um cenário de afluência, que alterará diretamente a quantidade de energia controlável, $EC(R,k)$, disponível em cada subsistema *R* para suprir a demanda do mercado.

Com relação aos problemas de diferentes períodos, as alterações nos problemas se darão diretamente nas energias armazenadas iniciais, que serão iguais às energias armazenadas finais obtidas na solução dos problemas do período imediatamente anterior e nas demandas do mercado. A combinação de diferentes valores de energias armazenadas e energias controláveis para suprir as diferentes demandas de mercado afetarão os resultados das gerações hidráulicas e térmicas, dos intercâmbios, e das energias vertidas e evaporadas, podendo causar eventual *deficit* de suprimento da demanda de mercado.

Todos os problemas do período *k-1* são resolvidos com os mesmos cortes obtidos no período *k*. Ao final da solução de todos os problemas do período *k-1*, novos cortes são gerados para serem utilizados no período *k-2*. Como estes cortes recém calculados já levam em consideração as conseqüências dos cortes do período *k-1*, somente eles precisam estar presentes nos problemas do período *k-2*, logo, numa mesma iteração, a quantidade máxima de cortes de um período para o anterior é igual à quantidade de cenários de afluências utilizada.

A função objetivo deste problema é composta da FCP e da FCF, logo, ela pode ser dividida da seguinte forma.

$$\alpha_{cen}^{1,B,k-1} = \underbrace{Min\left\{\sum_{R}\left[C_T\left(GT(R,k-1)\right)+C_D\left(DEF(R,k-1)\right)\right]\right\}}_{FCP} + \underbrace{\frac{1}{1+\beta}\alpha^{1,B,k}}_{FCF} \tag{59}$$

Denominando a FCP de $z_{cen}^{1,B,k-1}$, a função objetivo pode ser descrita como $\alpha_{cen}^{1,B,k-1} = z_{cen}^{1,B,k-1} + \frac{1}{1+\beta}\alpha^{1,B,k}$. Esta divisão das parcelas da FCP e FCF é muito importante para a determinação dos limites inferior e superior do processo iterativo, que serão calculados durante o ciclo *forward* que se inicia logo após o ciclo *backward*.

O ciclo *backward* prossegue com a solução sucessiva dos problemas de cada cenário, sempre construindo os cortes de Benders para ser utilizado no período anterior, até a solução dos problemas do segundo período de estudo. Os problemas do primeiro período não precisam ser resolvidos porque os cortes gerados por estes problemas não seriam aproveitados em nenhum problema.

- **Primeiro Ciclo da Solução *Forward***

No ciclo *forward* serão calculados os limites inferior e superior do processo iterativo de solução do problema de planejamento da operação.

Os problemas deste ciclo têm a formulação apresentada em (60).

$$\alpha_{cen}^{1,F,k} = Min\left\{\sum_{R}\left[C_T\left(GT(R,k)\right)+C_D\left(DEF(R,k)\right)\right]\right\} + \frac{1}{1+\beta}\alpha^{1,B,k+1}$$

Sujeito a (60)

$$EA(R,k+1) = EA(R,k) + EC(R,k) - GH(R,k) - EVERT(R,k) - EVAP(R,k) - EVMOR(R,k), \forall R$$

$$GH(R,k)+GT(R,k)+\sum_{s\in R}(INT(s,R,k)-INT(R,s,k))=DEM(R,k)-DEF(R,k),\ \forall R$$

$$\underline{GH}(R)\le GH(R,k)\le \overline{GH}(R),\forall R$$

$$\sum_{\forall s\neq R} INT(s,R,k)-\sum_{\forall s\neq R} INT(R,s,k)=0,\ \forall R$$

$$\underline{INT}(s,R)\le INT(s,R,k)\le \overline{INT}(s,R),\ \forall s\neq R,\ \forall R$$

$$\underline{GT}(R)\le GT(R,k)\le \overline{GT}(R),\forall R$$

$$\underline{EA}(R,k)\le EA(R,k)\le \overline{EA}(R,),\forall R$$

$$\alpha_1^{1,B,k+1}-\sum_R \pi_1^{1,k+1}EA_1^{1,k+1}(R,k+1)\cdot EA(R,k+1)\ge \delta_1^{1,k+1}$$
$$\alpha_2^{1,B,k+1}-\sum_R \pi_2^{1,k+1}EA_2^{1,k+1}(R,k+1)\cdot EA(R,k+1)\ge \delta_2^{1,k+1}$$
$$\vdots$$
$$\alpha_c^{1,B,k+1}-\sum_R \pi_c^{1,k+1}EA_c^{1,k+1}(R,k+1)\cdot EA(R,k+1)\ge \delta_c^{1,k+1}$$

Onde:

$\alpha_{cen}^{1,F,k}\rightarrow$ Valor esperado do custo de operação do período *k* para o cenário de afluência *cen* no primeiro ciclo *forward*.

$\alpha^{1,B,k+1}\rightarrow$ Variável escalar representando o valor esperado do custo de operação do período *k+1* no primeiro ciclo *backward*.

$\pi_{cen}^{1,k+1}EA_{cen}^{1,k+1}\rightarrow$ Derivada da função objetivo em relação à energia armazenada de um subsistema no período *k+1* da primeira iteração, para o cenário *cen*. A quantidade destas restrições é função do número do corte de Benders (cenário de afluência) e do número de iterações.

$\delta_{cen}^{1,k+1}\rightarrow$ Termo constante da restrição linear do corte de Benders.

Na solução do primeiro período, o problema de cada cenário possuirá as restrições dos cortes de Benders obtidos na solução do segundo período durante o ciclo *backward* imediatamente anterior, assim como a FCF, definida por $\frac{1}{1+\beta}\alpha^{1,B,2}$. Por definição, a FCF fornece o custo de operação do problema do período seguinte até o final do estudo. Logo, ao se obter o valor da FCP num determinado cenário do primeiro período, será obtido um custo de operação total com uma determinada probabilidade de ocorrência. Em outras palavras, a solução do problema $\alpha_{cen}^{1,F,1}=Min\left\{\sum_R[C_T(GT(R,1))+C_D(DEF(R,1))]\right\}+\frac{1}{1+\beta}\alpha^{1,B,2}$ fornecerá um custo de operação total para o cenário *cen* com uma probabilidade de ocorrência de p_{cen}.

A solução do problema de cada cenário é definida pelas seguintes equações: Em (61), tem-se a formulação original da função objetivo, que possui duas parcelas, redefinidas em (62) como FCP do cenário e FCF. Chamando a solução da FCP de cada cenário como $z_{cen}^{1,F,1}$, a função objetivo do problema ficará da forma descrita em (63) e a FCP será a função definida em (64).

$$\alpha_{cen}^{1,F,1} = Min\left\{\sum_{R}\left[C_T\left(GT(R,1)\right) + C_D\left(DEF(R,1)\right)\right]\right\} + \frac{1}{1+\beta}\alpha^{1,B,2} \tag{61}$$

$$\alpha_{cen}^{1,F,1} = FCP_{cen} + FCF \tag{62}$$

$$\alpha_{cen}^{1,F,1} = z_{cen}^{1,F,1} + \frac{1}{1+\beta}\alpha^{1,B,2} \tag{63}$$

$$z_{cen}^{1,F,1} = Min\left\{\sum_{R}\left[C_T\left(GT(R,1)\right) + C_D\left(DEF(R,1)\right)\right]\right\} \tag{64}$$

Onde:

$\alpha_{cen}^{1,F,1} \rightarrow$ Custo de operação do primeiro período para o cenário de afluência *cen* no primeiro ciclo *forward.*

$\alpha^{1,B,2} \rightarrow$ Valor escalar que representa o valor esperado do custo de operação do segundo período do primeiro ciclo *backward.*

$FCP_{cen} \rightarrow$ Função de custo presente para o cenário *cen*.

$FCF \rightarrow$ Função de custo futuro.

$z_{cen}^{1,F,1} \rightarrow$ Valor esperado da função de custo presente do cenário *cen* do primeiro período *forward*.

Tanto o custo total de operação quanto a FCP, α e z (63), respectivamente, estão definidos por cenário, logo, para a obtenção destes valores para cada período, é necessária a soma ponderada pela probabilidade de ocorrência de cada cenário, conforme definido a seguir.

$$\alpha^{1,F,1} = \frac{\sum_{cen}\alpha_{cen}^{1,F,1} p_{cen}}{NumCen} \tag{65}$$

$$z^{1,F,1} = \frac{\sum_{cen} z_{cen}^{1,F,1} p_{cen}}{NumCen} \qquad (66)$$

Onde:

$\alpha^{1,F,1} \rightarrow$ Valor esperado do custo de operação do primeiro período do primeiro ciclo *forward*.

$cen \rightarrow$ Cenário de energia afluente.

$\alpha_{cen}^{1,F,1} \rightarrow$ Valor esperado do custo de operação do cenário *cen* do primeiro período do primeiro ciclo *forward*.

$p_{cen} \rightarrow$ Probabilidade de ocorrência do cenário *cen*.

$NumCen \rightarrow$ Quantidade de cenários.

$z^{1,F,1} \rightarrow$ Valor esperado da FCP do primeiro período do primeiro ciclo *forward*.

$z_{cen}^{1,F,1} \rightarrow$ Valor esperado da FCP do cenário *cen* do primeiro período do primeiro ciclo *forward*.

O valor do custo de operação do primeiro período do ciclo *forward* fornece um limite inferior para o custo total de operação do estudo, podendo ser chamado de Z_{inf}.

O limite máximo do custo total de operação do estudo, chamado de Z_{sup}, será obtido através da soma dos custos presentes de todos os períodos de estudo, conforme está mostrado em (67).

$$z^{1,F} = \sum_{per} z^{1,F,per} \qquad (67)$$

Onde:

$per \rightarrow$ Períodos do estudo.

$z^{1,F,per} \rightarrow$ FCP do período *per* do primeiro ciclo *forward*.

$z^{1,F} \rightarrow$ Soma de todas as FCPs do primeiro ciclo *forward*.

Para a avaliação da convergência do processo iterativo de solução, será necessário comparar os limites Z_{inf} e Z_{sup}, mas como eles são variáveis estatísticas, a comparação tem que ser feita através de um intervalo de confiança e não através da pura comparação dos valores.

- **Demais Ciclos da Solução *Backward-Forward***

A seqüência dos cálculos dos problemas dos ciclos *backward/forward* deve continuar até que a diferença entre os valores de Zinf e Zsup estejam dentro de uma tolerância previamente especificada.

Convém destacar que a cada novo ciclo *backward/forward*, ou seja, a cada nova iteração, um novo conjunto de cortes, obtido nos problemas do ciclo *backward*, é adicionado aos problemas, resultando que o número máximo de cortes de um problema seja igual ao número de cenários multiplicado pelo número de iterações. Este aumento das restrições dos problemas faz com que o tempo computacional gasto na solução das últimas iterações seja bem maior do que o tempo gasto nas primeiras.

4.1.2 Estratégia Proposta de Paralelização

Ao longo do relato do detalhamento da solução seqüencial, podem-se notar as seguintes características existentes nos ciclos *backward* e *forward*:

1. Em cada período, a solução do PL de um cenário de afluência é independente da solução dos outros problemas, referente aos outros cenários de afluência.

   Uma vez que estes problemas são independentes entre si, a aplicação de técnicas de processamento paralelo faz com que eles possam ser resolvidos de forma simultânea em diversos processadores.
2. No ciclo *backward*, existe claramente um ponto de sincronismo que é a inclusão dos cortes de Benders, obtidos através dos problemas dos cenários de afluência resolvidos no período imediatamente anterior.

Para explicar a estratégia proposta de paralelização, serão apresentados diagramas dos ciclos *backward* e *forward*, onde estarão representados os cálculos na forma seqüencial, no lado esquerdo, e na estratégia de paralelização proposta no lado direito. Nestes diagramas, apenas para efeito de exemplificação, serão adotados quatro cenários e a estratégia de paralelização será representada com dois processadores.

Para melhor compreensão dos índices que são adotados nos diagramas, está mostrada na Figura 28 a convenção adotada para os índices que definem a iteração, período, cenário e ciclo dos cortes de Benders, dos valores esperados do custo de operação e da função de custo presente.

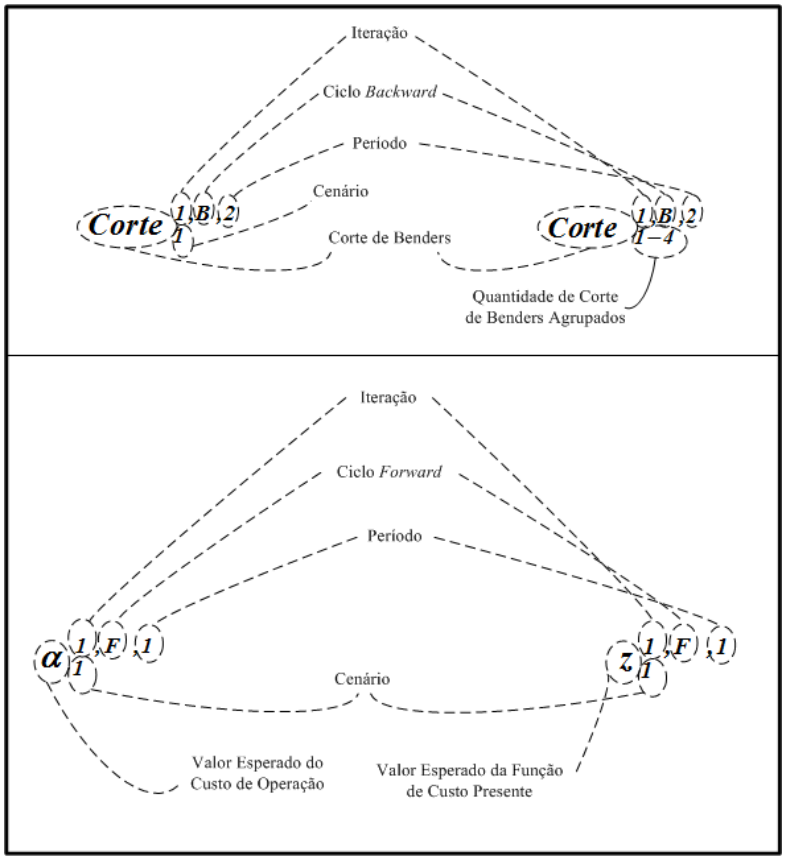

Figura 28 – Definição dos Índices das Variáveis

Na Figura 29 está mostrada a legenda utilizada na representação da solução dos diversos PLs. Em cada um destes problemas estarão representados o período e o cenário, além dos cortes utilizados na formulação do mesmo, e destacadas determinadas variáveis de saída com o objetivo de facilitar o entendimento da estratégia de paralelização.

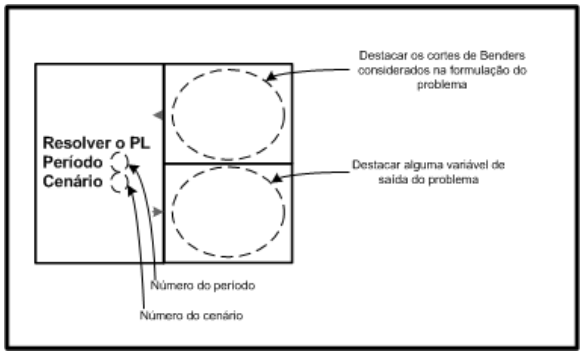

Figura 29 – Legenda Utilizada na Solução dos PLs

Na Figura 30 está mostrada a legenda utilizada na representação do agrupamento dos cortes. Os cortes a serem agrupados estão destacados acima do retângulo, que representa uma rotina ou função responsável pelo agrupamento dos mesmos. Abaixo deste retângulo está mostrado o conjunto de cortes que foram disponibilizados. Neste trabalho, o ato de agrupar um conjunto de cortes significa a simples inclusão dos mesmos numa base de dados para posterior disponibilização. O significado da variável $Corte_{1\text{-}4}$ é que quatro cortes, agrupados num determinado período, estão sendo disponibilizados para serem levados em consideração na formulação de um problema.

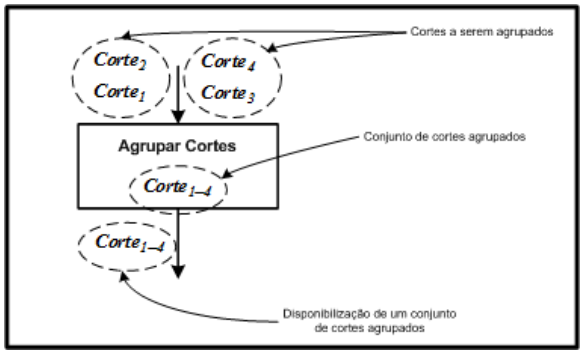

Figura 30 – Legenda Utilizada nos Agrupamentos de Cortes

- **Primeira Iteração do Ciclo *Backward***

O primeiro período a ser resolvido no ciclo *backward* é o último, representado pelo período *k*. Neste período não existe cortes representativo da FCF e a solução dos quatro cenários fornece quatro cortes que são disponibilizados para os problemas do período *k-1*. O processamento seqüencial para a realização destes cálculos está mostrado no lado esquerdo da Figura 31.

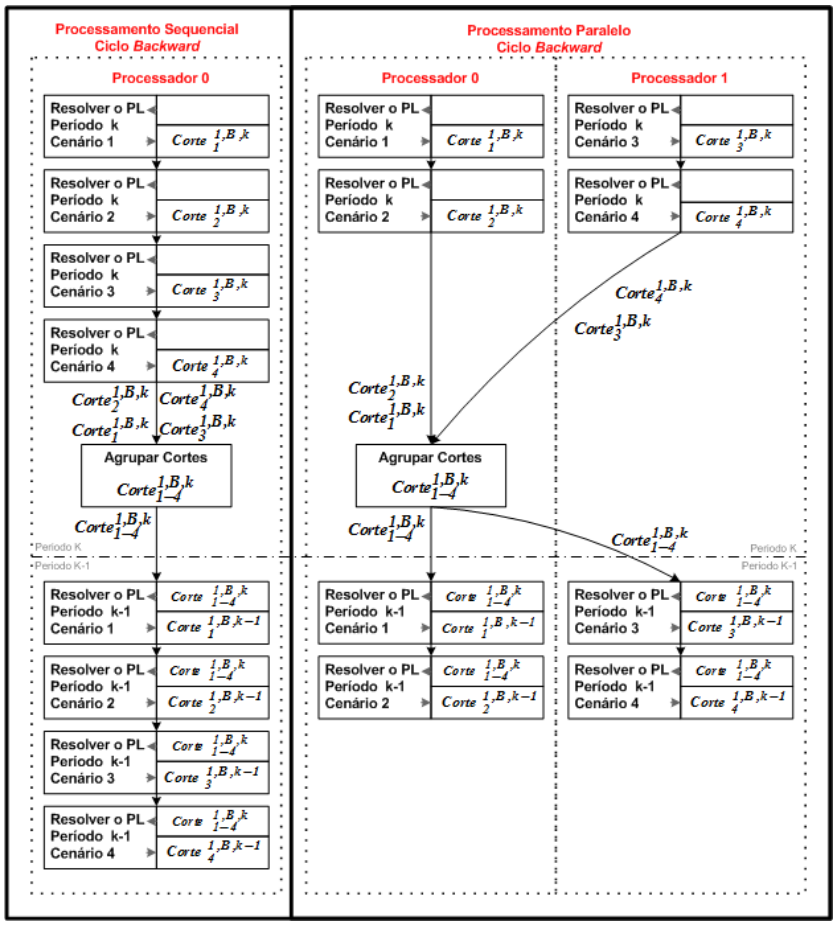

Figura 31 – Diagrama da Primeira Iteração do Ciclo *Backward*

A estratégia de paralelização proposta é separar os PLs entre os processadores participantes do ambiente de processamento paralelo. No diagrama da Figura 31 pode-se observar a solução de dois PLs em cada um dos dois processadores. Após a solução destes PLs, é necessário o envio de todos os cortes que não estão disponíveis no processador 0, para que ele possa agrupá-los e enviá-los para todos os outros processadores. O principal objetivo do ciclo *backward* é determinar os cortes de Benders para os períodos de estudo. Cada problema resolvido com um cenário de afluência gera um corte de Benders e todos eles são disponibilizados para todos os processadores em forma de restrições para o próximo problema a ser resolvido.

Na solução do período *k-1*, todos os cortes gerados no período *k* devem estar disponíveis para serem levados em consideração na formulação dos PLs. Já os cortes produzidos por estes PLs deverão ser agrupados e disponibilizados para todos os

problemas do período *k-2*, que serão resolvidos a seguir. Este processo é repetido até chegar ao período dois, onde serão calculados os cortes para o período 1, ou seja, no ciclo *backward*, o problema do primeiro período não é executado.

- **Primeira Iteração do Ciclo *Forward***

O ciclo *forward* inicia-se com a solução dos PLs referentes ao primeiro período de estudo. Nestes PLs deverão ser levados em consideração, nas respectivas formulações, os cortes produzidos na solução dos PLs do período dois no ciclo *backward*. Convém ressaltar que todos os cortes utilizados no ciclo *forward* foram obtidos durante a passagem pelo ciclo *backward* imediatamente anterior. Logo, teoricamente, todos eles estariam disponíveis para todos os processadores, não sendo, portanto, necessário os reenvios dos mesmos em cada início de período. Porém, o armazenamento de todos os cortes ocuparia uma grande área de memória, que talvez não esteja disponível durante a implementação desta estratégia de paralelização. Por causa disto, foi colocado no diagrama, mostrado na Figura 32, uma etapa no início de cada período representando a disponibilização do conjunto de cortes para todos os processadores. Em teoria, este ponto de sincronismo não deveria existir, porém, na prática, possivelmente, existirá.

A estratégia de paralelização é, basicamente, a mesma que foi adotada no ciclo *backward*. Cada processador fica responsável em resolver um conjunto de PLs. No exemplo mostrado pela Figura 32, pode-se observar a solução dos problemas dos cenários um e dois no primeiro processador e dos cenários três e quatro no segundo processador. Os mesmos quatro cenários são resolvidos de forma seqüencial na parte esquerda do diagrama.

Deve-se observar que na solução de cada um dos PLs do primeiro período de estudo, as variáveis z e α são disponibilizadas. Estas variáveis serão utilizadas na verificação da convergência que se dará ao final deste ciclo *forward*. Os cálculos necessários para a determinação da convergência do processo iterativo serão realizados no processador 0, logo, estas variáveis deverão estar disponíveis neste processador. Com o objetivo de não criar pontos de sincronismos ao longo da solução do ciclo *forward*, foi decidido enviar estas variáveis somente após as soluções dos PLs do último estágio, conforme está mostrado no diagrama da Figura 33.

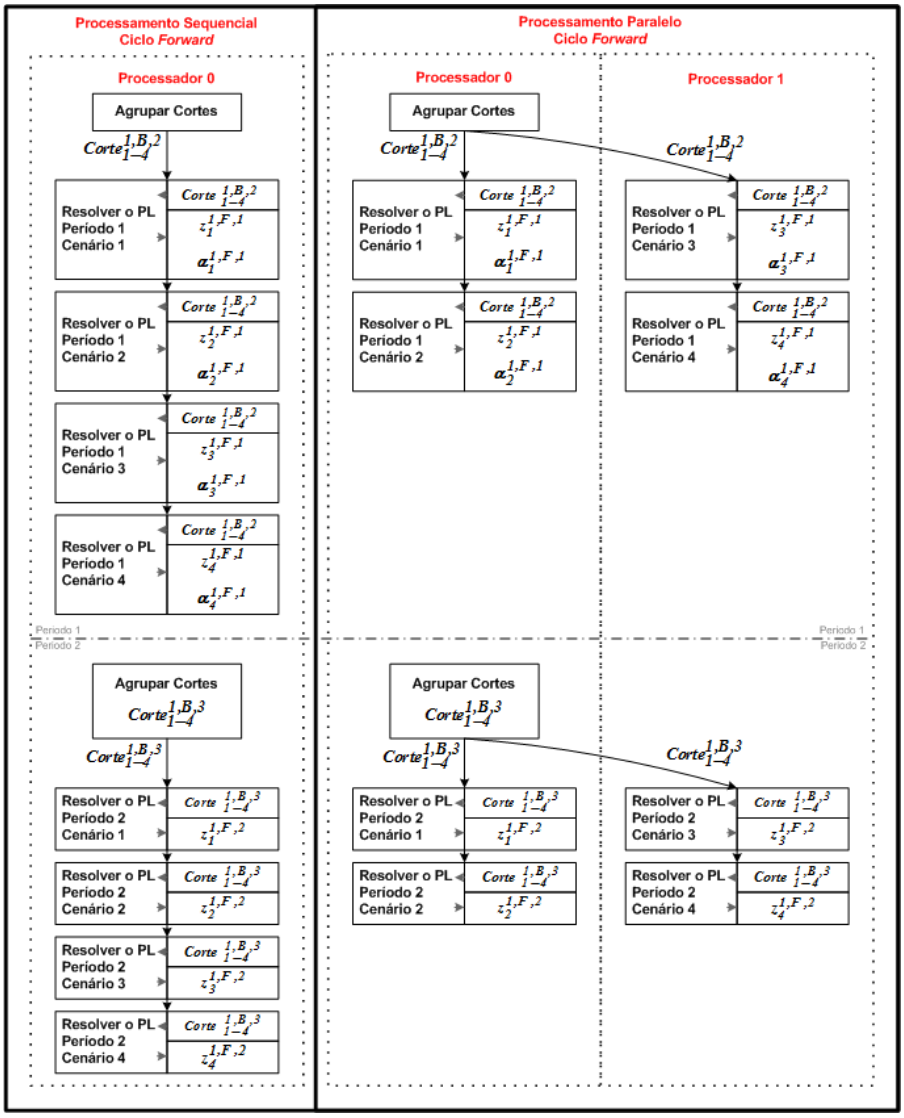

Figura 32 - Diagrama da Primeira Iteração do Ciclo *Forward*

Após os cálculos do último período, todas as variáveis z e α são enviadas para o processador 0, que calculará o custo total de operação esperado do primeiro período $\alpha^{1,F,1}$ e comparará com o somatório de todos os custos presentes de todos os períodos $z^{1,F}$. A convergência será obtida quando a diferença entre estas duas variáveis seja menor que uma tolerância pré-estabelecida.

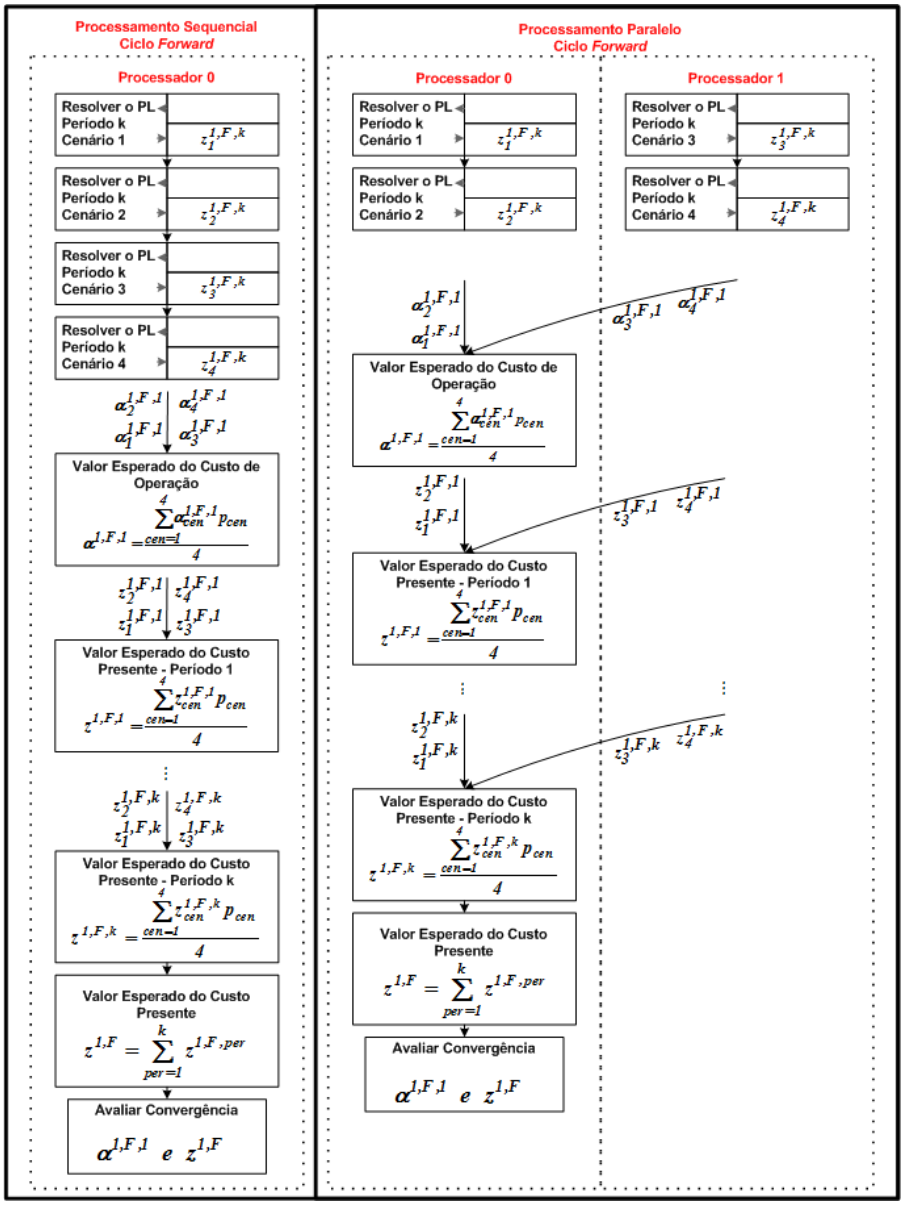

Figura 33 - Diagrama do Último Período da Primeira Iteração do Ciclo *Forward*

- **Segunda Iteração do Ciclo *Backward***

Caso a convergência não seja obtida, uma nova iteração deverá ser executada. Os cálculos desta segunda iteração são similares aos da primeira, conforme está mostrado no diagrama da Figura 34.

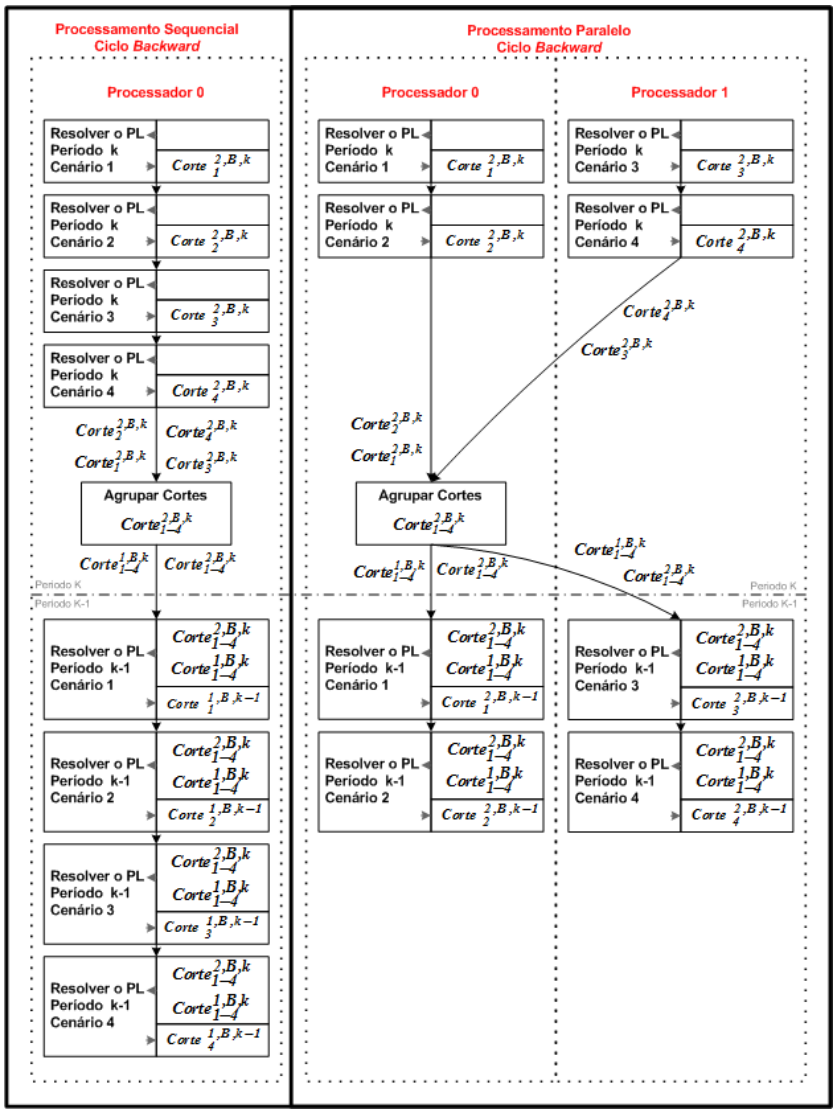

Figura 34 - Diagrama da Segunda Iteração do Ciclo *Backward*

Neste ciclo, deve-se destacar que, a partir do período *k-1*, os cortes a serem levados em consideração na formulação dos PLs, não serão apenas os que acabaram de ser calculados e disponibilizados nesta segunda iteração, mas também àqueles obtidos na iteração anterior. Por conta deste fato, percebe-se que, quanto maior for a quantidade de iterações necessárias para a convergência do processo, maiores serão os PLs a serem resolvidos e maiores serão os tempos gastos nestas soluções, tornando fundamental o processamento paralelo para encontrar a solução num tempo considerado razoável.

- **Segunda Iteração do Ciclo *Forward***

O ciclo *forward* da segunda iteração está mostrado nos diagramas da Figura 35 e da Figura 36.

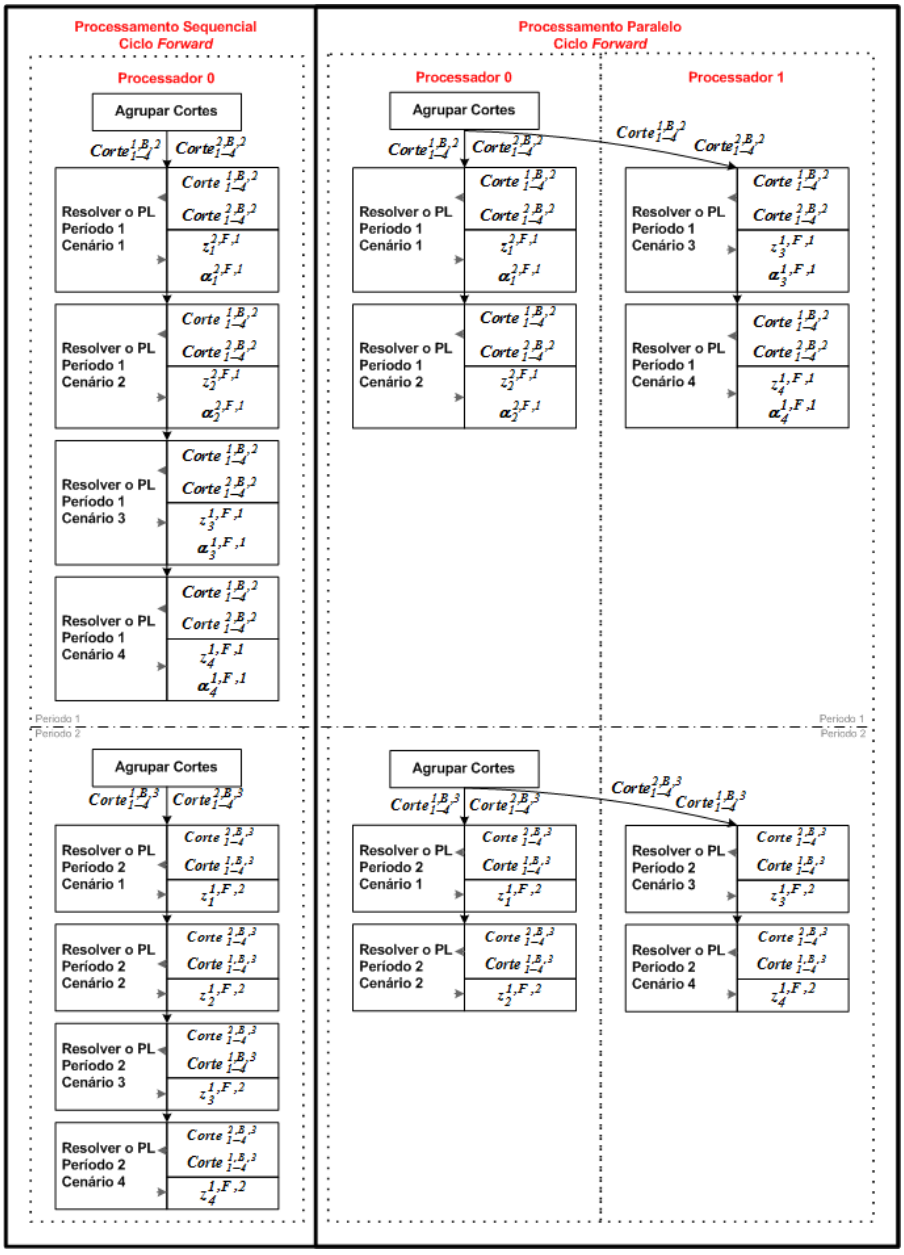

Figura 35 - Diagrama da Segunda Iteração do Ciclo *Forward*

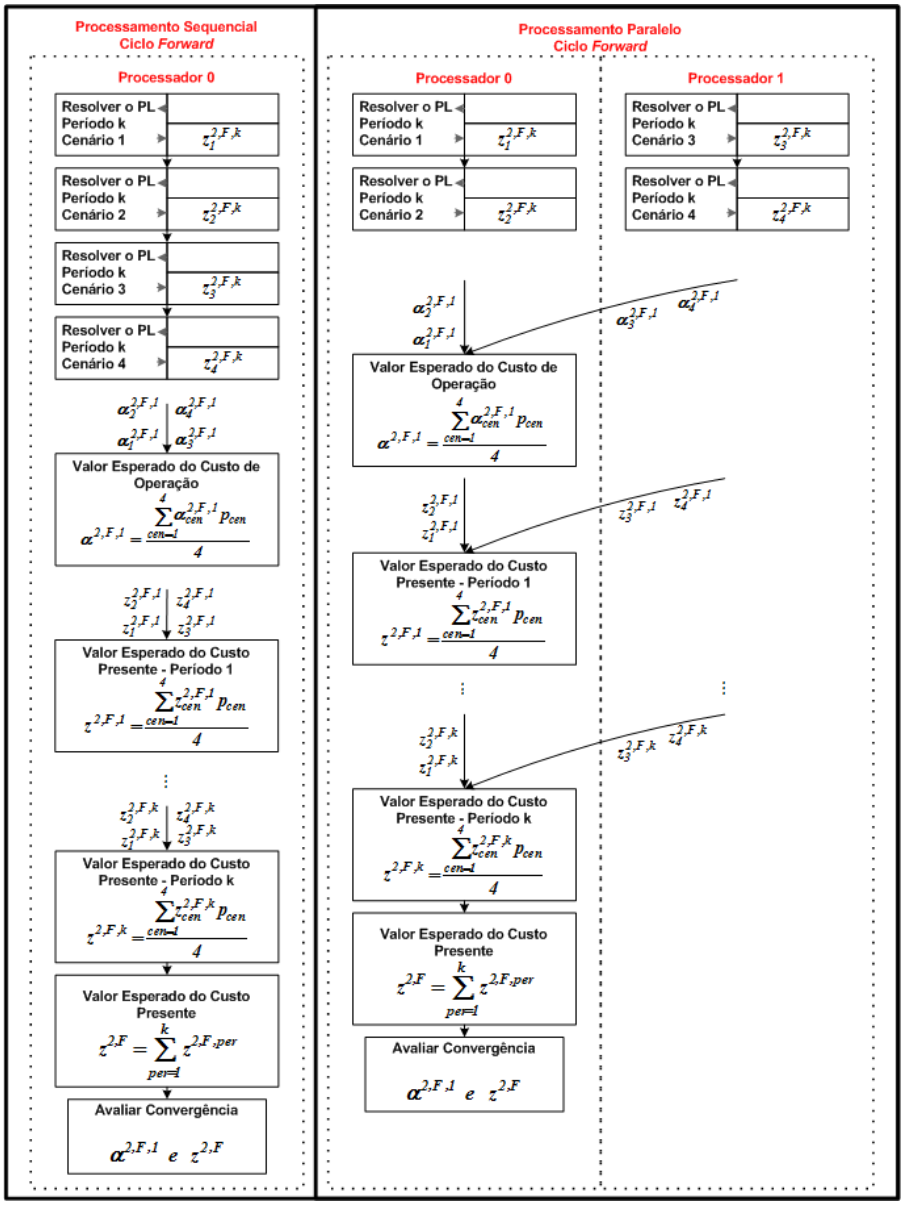

Figura 36 - Diagrama do Último Período da Segunda Iteração do Ciclo *Forward*

Os mesmos comentários feitos na primeira iteração valem para os cálculos desta segunda iteração. O importante a destacar neste ciclo da segunda iteração é o aumento da quantidade de cortes a serem levados em consideração na formulação dos problemas. Por causa disto, os problemas deste ciclo, tal qual ocorrido no segundo ciclo *backward*, também aumentam de tamanho e de tempo de solução.

- **Demais Iterações**

Este processo iterativo seguirá até que a convergência seja obtida. A estratégia de paralelização adotada nestas iterações é exatamente a mesma que foi apresentada para as suas primeiras.

4.2 Aplicação da Metodologia Paralela no Modelo NEWAVE

O modelo NEWAVE, desenvolvido no CEPEL, é atualmente utilizado pelo Operador Nacional do Sistema Elétrico (ONS), no planejamento da operação energética de médio prazo, e pelo Ministério de Minas e Energia (MME) e pela Empresa de Pesquisa Energética (EPE), nos estudos do planejamento da expansão. O objetivo básico destes planejamentos é determinar as metas de geração para cada usina, que atendam a demanda e minimizem o valor esperado do custo de operação ao longo do período de planejamento. Este custo é composto pelo custo variável de combustível das usinas térmicas e pelo custo atribuído às interrupções de fornecimento de energia, representado por uma função de penalização das faltas de energia (custo do *deficit*).

A política de operação depende dos cenários de operação futuros, sendo que alguns parâmetros que definem estes cenários estão apresentados a seguir:

- Afluências hidrológicas;
- Demanda dos submercados;
- Preços dos combustíveis;
- Custos de *deficit*;
- Planejamento de novas usinas;
- Disponibilidade dos geradores, linhas de transmissão e transformadores que impactam a transmissão de grandes blocos de energias entre os sistemas.

A previsão destes parâmetros é muito complexa e sujeita a um nível elevado de incerteza. As incertezas podem ser representadas de duas formas básicas: forma explícita – quando a distribuição de probabilidades do parâmetro é representada diretamente no cálculo da política de operação; e forma implícita – quando a incerteza do parâmetro é representada através de análise de sensibilidade ou através da utilização de valores médios.

No sistema hidroelétrico brasileiro, os reservatórios possuem capacidade de regularização plurianual e os períodos secos, nos quais as afluências são baixas, podem apresentar duração de até alguns anos. Logo, é fundamental representar de forma precisa o efeito da estocasticidade das afluências no planejamento da operação de longo prazo. Segundo o manual, o modelo NEWAVE representa de forma explicita somente a incerteza associada à hidrologia. As demais incertezas são representadas de forma implícita.

A metodologia de paralelização, aqui proposta, foi aplicada na versão 15 do modelo NEWAVE. Em relação à notação utilizada na teoria da metodologia paralela proposta, mostrada na Figura 27, a única diferença é o conceito de abertura de cenários de afluência, utilizado no ciclo *backward*, conforme está mostrado na Figura 37. O objetivo da utilização de vários cenários de afluência, em vez de apenas um, é melhorar a representatividade dos cortes de Benders gerados neste ciclo. Todos os outros conceitos se aplicam tanto ao ciclo *backward* quanto ao ciclo *forward*.

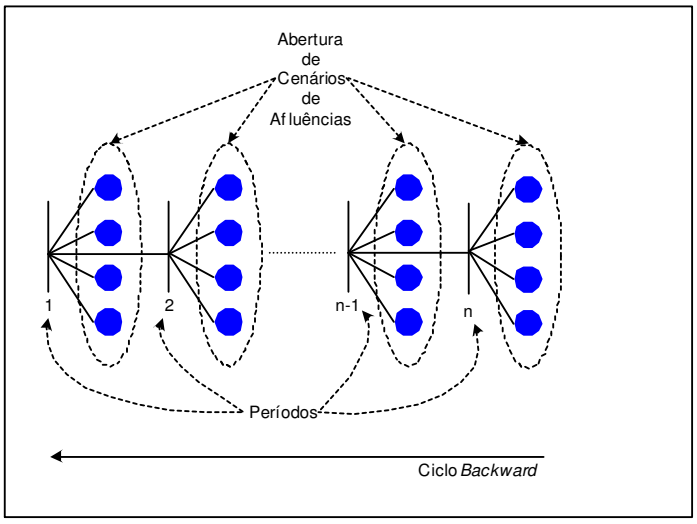

Figura 37 - Definição de Parâmetros Utilizados no Ciclo *Backward* da Aplicação

No processo *backward*, em cada estágio *t* do horizonte de planejamento, cada processador recebe um conjunto de problemas de despacho de operação associados a diferentes cenários de energias afluentes. De posse das soluções de cada um dos cenários, cada processador gera um corte de Benders que é enviado para o processador mestre. Esse processador, de posse de todos os cortes de Benders recebidos, constrói a

FCF que será utilizada no estágio *t-1* e a transmite para todos os processadores participantes do cálculo, conforme pode ser visto na Figura 38.

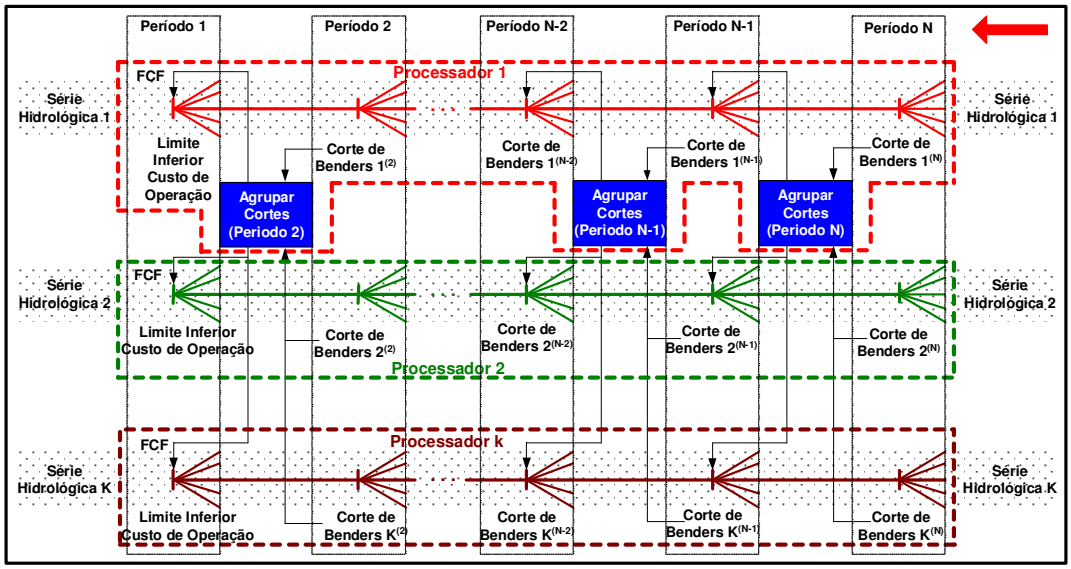

Figura 38 – Distribuição dos Cortes de Benders no Processo *Backward*

De forma semelhante, no processo *forward*, em cada estágio, os processadores recebem um conjunto de problemas de PL, cujas soluções são transferidas para o processador mestre para verificação da convergência. Com relação aos cortes, como pode ser visto na Figura 39, em cada período de execução o processador mestre envia os valores necessários para que os mesmos cortes sejam levados em consideração em todos os problemas. Caso a convergência não tenha sido alcançada, uma nova iteração é executada, de forma distribuída, até a convergência final.

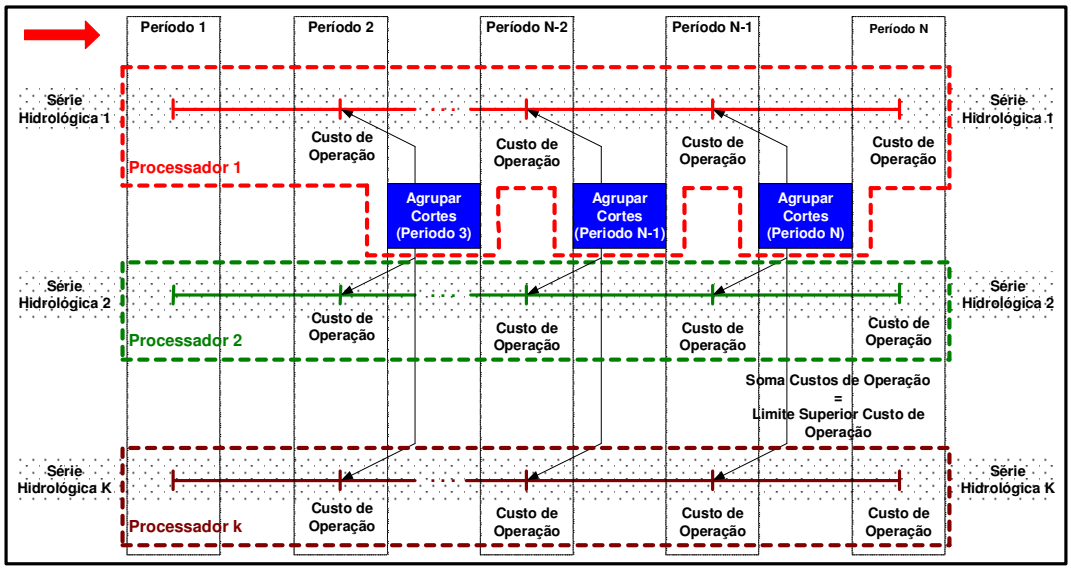

Figura 39 – Distribuição dos Cortes de Benders no Processo *Forward*

Com relação à divisão dos problemas entre os diversos processadores, esta foi feita através da divisão da quantidade de simulações pela quantidade de processadores. No ciclo *backward*, o processador que receber determinado cenário fica responsável também por todos os PLs referentes às diferentes aberturas. Este modo de divisão das simulações não leva em consideração uma possível heterogeneidade dos processadores nem os diferentes tempos de solução de cada problema, o que pode levar à ociosidade de alguns processadores que terminem mais rapidamente de resolver o seu conjunto de problemas.

No anexo A estão descritas detalhadamente diversas tarefas executadas para implementar a metodologia proposta no modelo NEWAVE.

4.3 Desempenho da Estratégia de Paralelização Inicial

O termo melhorar o desempenho tem por objetivo diminuir o tempo de execução de uma aplicação computacional. Logo, é necessário dispor de alguma ferramenta que permita a obtenção dos tempos gastos na execução das tarefas do programa e este procedimento é comumente chamado de perfilagem do código do programa.

No caso de uma aplicação paralela é possível dividir o processo de perfilagem do código fonte em duas etapas: (1) perfilagem serial; e (2) perfilagem paralela.

A perfilagem serial consiste na otimização do código propriamente dito, alterando, por exemplo, instruções de execução mais lenta por outras de execução mais rápida. No sistema operacional Linux pode ser utilizado o perfilador *gprof*, que é incorporado ao compilador GNU Fortran e normalmente está disponível em qualquer distribuição deste sistema operacional.

Já a perfilagem paralela consiste na otimização da estratégia de paralelização utilizada, identificando e eliminando, por exemplo, pontos de contenção, melhorando a distribuição de carga entre os processadores etc.

Tanto na perfilagem paralela quanto na serial adota-se um procedimento cíclico para melhorar o desempenho da aplicação. Neste ciclo, conforme está mostrado na Figura 40, o primeiro passo é obter informações sobre os tempos de processamento necessários para que as tarefas sejam executadas. De posse destes tempos, uma análise detalhada é necessária para identificar quais são as tarefas que demandam muito esforço computacional e quais são as melhores maneiras de diminuir estes esforços. Nesta análise também podem ser identificados alguns problemas de esforços exagerados para a solução de determinadas tarefas, que deverão ser corrigidos. A escolha das alternativas para executar as mesmas tarefas de forma mais eficiente é um processo de suma importância, que deve ser executado com muito cuidado e atenção para que se obtenha sucesso na melhoria do desempenho de um programa. Com a identificação dos problemas e as soluções determinadas, o passo seguinte é implementar as mudanças no código do programa, de forma a se obter novos resultados de desempenho e se iniciar um novo ciclo.

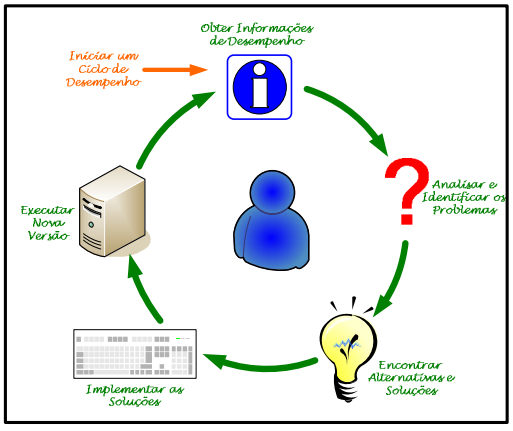

Figura 40 – Etapas Cíclicas para Otimização do Desempenho de um Programa Computacional

Uma importante observação a ser feita é que, excetuando-se o item 4.3.1.1, no qual a palavra processador designa o processador propriamente dito, atualmente composto de vários núcleos de processamento, ao longo deste trabalho quando o termo processador for utilizado, salvo indicação contrária, estará se fazendo referência ao núcleo de processamento.

4.3.1 Características do Ambiente de Processamento

Todos os resultados foram obtidos através da execução da versão 15 do programa NEWAVE, modificado com a inclusão da estratégia de paralelização proposta, num *cluster* do tipo *blade* com capacidade máxima de 336 núcleos de processamento, além de um servidor responsável pelo gerenciamento da máquina. Nos itens a seguir serão apresentadas as configurações dos processadores, assim como os programas básicos que estão instalados nas máquinas.

4.3.1.1 Configuração dos Processadores

O servidor responsável pelo gerenciamento do *cluster* tem as seguintes características principais: modelo IBM x3550 com processador INTEL Xeon, modelo E5405, com quatro núcleos de processamento (*quad-core*), com freqüência do processador de 2,00GHz, tecnologia EM64T; freqüência do barramento de memória (FSB) de 1.333MHz; memória cache L2 interna de 12Mbytes; memória RAM de 2Gbytes; dois discos rígidos de 73Gbytes SAS de 15.000rpm configurados em RAID-1, *hot-swap*; e duas placas de rede 10/100/1000Mbps.

O processamento dos casos é feito por um conjunto de 42 placas do tipo *blade*, cada uma com 2 processadores com 4 núcleos de processamento (*quad-core*), totalizando 84 processadores e 336 núcleos de processamento. Todos os processadores são do tipo Xeon, da Intel, *Core2Quad* modelo X5355, com freqüência do processador de 2,66Mhz, a freqüência do barramento de memória (FSB) é de 1.333MHz, o tamanho da memória cache L2 interna é de 8MBytes, o disco rígido local de 136GBytes, do tipo SAS com velocidade de rotação de 10.000rpm e com duas placas de rede de 10/100/1000Mbps.

Para armazenamento de dados e resultados, o sistema possui um *storage* com 11 discos de 500GBytes, *hot swap*, do tipo SATA, operando em RAID-5, totalizando uma capacidade útil de armazenamento de 4,1TBytes.

4.3.1.2 Configuração dos Programas

Os seguintes programas principais estão instalados e configurados:

- Sistema operacional Linux CentOS versão 5.2, 64 bits, baseado na distribuição RedHat Enterprise Linux;
- Biblioteca MPICH2 para troca de mensagens do MPI, versão 1.0.8;
- Sistema de fila Torque, baseado no OpenPBS, que tem por finalidade alocar os recursos computacionais no *cluster* para o processamento dos programas e permite o gerenciamento de filas
- Compiladores GNU C, Fortran77 e Fortran90.

4.3.2 Caso Utilizado

Para avaliar o desempenho da estratégia de paralelização proposta foi escolhido o caso do Programa Mensal da Operação (PMO) de Março de 2009, disponibilizado pelo ONS. Este caso possui as seguintes características:

- Período de estudo de 5 anos e período pós-estudo de 5 anos, totalizando 120 meses;
- Número de cenários de afluências = 200;
- Número de aberturas no ciclo *backward* = 20;
- Número de cenários de afluências na simulação final = 2000.

4.3.3 Perfilagem Serial

A perfilagem serial consiste na determinação do perfil dos tempos gastos nas instruções que compõem um programa computacional. Para a obtenção destes tempos, utilizou-se o utilitário GNU *gprof*, que é um aplicativo disponível nas distribuições Linux. A execução com 1 processador da versão do programa NEWAVE apresentou os resultados mostrados na Tabela 2. Nesta tabela estão apresentadas as seis rotinas que mais consumiram tempo no processo de perfilagem, a soma dos tempos médios destas

seis rotinas, o tempo médio total da perfilagem e o tempo médio total de execução dos casos.

Tabela 2 – Tempos Médios Gastos no Processamento

Rotinas	Tempo Médio (s)
Alterar_Vetor_b	1.180,42
Salvar_Solução	475,94
Montar_Vetor_b	322,12
Calcular_Pi_Água	192,89
Montar_Cortes	162,62
Alterar_Vetor_b_Inicial	137,63
Soma Tempos Rotinas	2.471,61
Tempo Total Perfilagem	2.612,32
Tempo Total Execução	89.702,67

É importante notar que o tempo total disponível na perfilagem foi de apenas 2,91% (2.612,32/89.702,67), indicando que a maior parte do tempo o processamento não está sendo feito nas rotinas do programa e sim na biblioteca de solução de PLs. Para a solução destes problemas, o programa NEWAVE utiliza o pacote comercial OSL®, *Optimization Subroutine Library*, desenvolvido pela IBM®. Esta biblioteca não possui código fonte aberto, logo, não é possível realizar a sua perfilagem e, como mostraram os resultados, cerca de 97% do tempo de processamento é consumido em cálculos internos feitos nela.

Com relação aos quase 3% do tempo perfilável, a Tabela 2 mostra que o consumo das seis rotinas apresentadas corresponde a 94,6% (2.471,61/2.612,32). Logo, a análise destas rotinas implica em que quase 95% do tempo de processamento perfilável do programa seja analisado.

É importante ressaltar que o método de obtenção destes tempos consiste em fazer amostras periódicas e verificar em que rotina ou, dependendo da opção desejada, em que linha da rotina, está o processamento. Logo, isso pode gerar informações não acuradas, uma vez que, dependendo do período de amostragem e o tempo de execução da instrução, ora a perfilagem pode informar que a instrução foi executada e ora não. Pelo manual do programa *gprof*, é informado que o período de amostragem é de 0,01s e para uma medida de n amostragens, o erro esperado no tempo informado é de $\sqrt{n}$. Logo, por exemplo, se o processo de perfilagem informar um tempo consumido de 1s, o número de amostragens foi 100 ($1/0,01$) e o erro esperado foi de 10 amostras ($\sqrt{100}$), o

que dá 0,1s ($10*0,01$) e que corresponde a 10% da medida. Já uma informação de tempo consumido de 100s, que corresponderia a 10.000 amostras ($100/0,01$), o erro esperado seria de 100 amostras ($\sqrt{10.000}$), que corresponde a 1s ($100*0,01$), que é equivalente a 1% da medida total. Estes exemplos mostram que o erro do tempo medido pode variar muito e que, quanto menor for o tempo informado, maior é o erro desta informação.

Outro cuidado que se deve ter em mente é que a perfilagem introduz alguma lentidão no processo original, logo, uma versão compilada e executada com o objetivo de perfilagem tende a ser mais lenta do que a mesma versão compilada para execução normal, devendo isso ser levado em consideração caso o tempo total de execução destas duas versões sejam comparados.

Por fim, as instruções que não puderem ser perfiladas não serão informadas no relatório do programa de perfilagem. Desde modo, todo o tempo gasto na execução das funções da biblioteca de solução de PLs não será levado em consideração, assim como as instruções de leitura e escrita em arquivo que não estiverem mais no âmbito do programa, ou seja, que estejam dependendo do sistema operacional para serem finalizadas.

4.3.3.1 Análise da Rotina que Altera o Vetor de Termos Independentes

A rotina Alterar_Vetor_b executa as atualizações do vetor de termos independentes a partir das mudanças nos valores das afluências de cada abertura de cada mês do período do caso executado e, pelo relatório de perfilagem, é responsável por 45% do tempo perfilável. Por conta deste expressivo consumo relativo de tempo de processamento, está será a primeira rotina a ser analisada.

Os resultados dos tempos de processamento por linha de código mostram que as instruções que mais consumiram tempo foram as equações dos cortes de Benders e a inicialização de um vetor auxiliar.

Analisando as operações referentes às equações dos cortes de Benders, chega-se à conclusão que realmente elas têm que consumir bastante tempo de processamento, uma vez que são utilizados dados de todos os cortes em cada abertura de cada série hidrológica de cada período. A cada iteração a quantidade de cortes aumenta, fazendo com que o consumo de tempo destas operações aumente significantemente com o

aumento da quantidade de iterações do caso. Uma possibilidade seria uma forma de armazenar os resultados para os cortes das iterações anteriores, uma vez que o resultado de um determinado corte é sempre o mesmo. O problema é que a quantidade de memória necessária para armazenar estes resultados é muito grande. Para se ter uma idéia, para o caso base, PMO de Março de 2009, seriam necessários 1.310.400.000 registros de 12 bytes cada, por conta das 15 iterações, 117 períodos, 200 séries, 20 aberturas e 200 cortes por iteração. A implementação deste armazenamento não seria uma tarefa simples e com a possibilidade do acesso a estes quase 16GBytes resultar num consumo de processamento maior que o tempo gasto atualmente, mesmo com a redução da quantidade de contas. Como o peso do tempo de processamento desta rotina é pequeno no tempo total de execução do caso, decidiu-se pela não execução desta forma de realizar estas operações dos cortes de Benders.

Com relação à inicialização do vetor auxiliar, percebe-se imediatamente um excesso que é a inicialização de todo o vetor com zero, enquanto que seria necessário apenas inicializar as posições que serão efetivamente utilizadas. Como a diferença entre estes dois valores é muito grande e esta rotina é muitas vezes chamada (6.669.000 vezes no caso base), a quantidade de operações desnecessárias é muito grande e a redução desta inicialização certamente fará com que o tempo gasto nesta rotina reduza de forma significativa. Numa análise mais detalhada deste vetor, chegou-se a conclusão que apenas poucas posições precisam ser inicializadas com zero e estas podem ser feitas logo depois da utilização do vetor auxiliar, fazendo com que toda a inicialização inicial fosse eliminada.

Além destas alterações, a análise das instruções desta rotina mostrou que outras operações poderiam ser otimizadas, tais como:

- Diminuição da quantidade de operações → existe uma correção de energia por um fator para cada submercado que era executada várias vezes ao longo da rotina. A solução foi realizar a correção da energia em cada submercado no início da rotina uma única vez e utilizar este resultado todas as outras vezes;

- Eliminação de operações intermediárias desnecessárias → Na rotina original existia muitas operações de atribuição num vetor auxiliar, para depois ocorrer a atribuição no vetor que efetivamente seria utilizado nos cálculos. Esta operação intermediária

foi eliminada de boa parte da rotina, sendo que as contas passaram a ser feitas diretamente no vetor definitivo, sem nenhuma perda de precisão nas contas. Somente em algumas situações realmente necessárias é que foi utilizado o vetor auxiliar.

Depois destas modificações na rotina, novos resultados foram obtidos e o tempo médio obtido pela perfilação caiu aproximadamente 33,5%, conforme está mostrado na Tabela 3.

Tabela 3 – Diferenças de Tempo da Rotina para Alterar o Vetor de Termos Independentes

Rotina Alterar_Vetor_b	Tempo Médio (s)
Antes	1180,42
Depois	785,24
Diferença (%)	-33,48

4.3.3.2 Análise da Rotina que Salva a Solução dos PLs

A segunda rotina a ser analisada é a responsável pela transferência da solução dos PLs da biblioteca de solução para as variáveis do programa. O resultado da perfilagem apontou que 18% do tempo perfilável foi consumido na rotina Salvar_Solução.

A análise da perfilagem das linhas do código fonte da rotina mostrou que, dentre as principais instruções consumidoras de tempo, existiam algumas duplicações no armazenamento da base viável do problema, logo, a eliminação destas duplicações fará com que ocorra uma diminuição do tempo final de processamento da rotina. Outro ponto é que o armazenamento da base era feito em diversos pontos da rotina, sempre com um teste condicional para saber se ela deveria ser armazenada ou não, fazendo com que o teste condicional fosse feito diversas vezes. Ao se agrupar as instruções para armazenamento da base, o teste condicional pode ser feito apenas uma vez, reduzindo um pouco mais o tempo gasto no processamento da rotina.

Os novos resultados, resultante das modificações propostas, mostram que o tempo perfilado foi reduzido em pouco mais de 15%, uma vez que o valor médio caiu de 475,94s para 403,04s, conforme está mostrado na Tabela 4.

Tabela 4 – Diferenças de Tempo da Rotina que Salva a Solução dos PLs

Rotina Salvar_Solução	Tempo Médio (s)
Antes	475,94
Depois	403,04
Diferença (%)	-15,32

4.3.3.3 Análise da Rotina que Monta o Vetor de Termos Independentes

A terceira rotina com maior tempo consumido é a responsável pela montagem do vetor de termos independentes. Com cerca de 12% do tempo total de perfilagem, esta rotina é extremamente simples, possuindo apenas as instruções necessárias para calcular os valores de cada posição do referido vetor. Estas operações são feitas através da multiplicação do valor de cada corte com as afluências históricas da série hidrológica corrente.

Uma vez que os cortes de uma iteração são compostos por cortes novos que foram recém incluídos com outros mais antigos, onde estas operações já foram feitas, seria possível armazenar o resultado final das operações com a parcela antiga dos cortes e acessá-las sempre que for necessário, diminuindo as operações de multiplicação somente aos cortes recém incluídos. Como o armazenamento destes resultados envolve uma grande quantidade de memória, uma vez que para cada corte é necessário guardar um resultado de 8 bytes para cada série hidrológica e para cada período do estudo, podendo chegar a 560MBytes para o caso base, com 200 séries hidrológicas, 15 iterações, 3000 cortes armazenados e 117 meses de estudo, fica inviável o uso de memória RAM. Diante disto, uma possível solução seria o armazenamento em disco, num arquivo de acesso direto para agilizar o acesso aos dados. Só que este tipo de arquivo necessita do armazenamento de um apontador, fazendo com que a quantidade total armazenada aumente para algo em torno de 840MBytes.

Como esta rotina é chamada 821.000 vezes ao longo da solução do caso base e levou em média 322s para executar todas estas operações, 0,392ms por acesso, é muito difícil que a substituição de boa parte destas operações pelo armazenamento em disco, com evolução crescente do tempo de acesso, por conta do aumento significativo do tamanho do arquivo, consiga ser mais eficiente que a estrutura atual da rotina. Outro agravante na execução multiprocessada é que um mesmo processador, por conta do

gerenciamento dinâmico da carga entre os processadores, não necessariamente resolve a mesma série hidrológica em diferentes iterações. Por conta disto, os resultados para todas as séries deverão ser agrupados e disponibilizados para todos os processadores, necessitando de mais tempo de processamento para realizar estas operações.

A implementação de somente a parte referente ao armazenamento dos totais para cada corte permite a execução com apenas 1 processador, mas o suficiente para avaliar se a opção pelo armazenamento é vantajosa ou não perante a solução inicial. Esta versão executou o caso base em aproximadamente 10 minutos a mais que a versão sem nenhuma modificação, indicando que é mais eficiente manter a rotina na sua forma original.

4.3.3.4 Análise da Rotina Calcula o Valor do π da Água

A perfilagem desta rotina mostrou que as instruções que mais consumiram tempo foram as que estavam relacionadas com a derivada da função de custo futuro. Porém, numa análise mais detalhada destas operações, percebe-se que o programa estava fazendo a conta para todos os cortes, porém, só seriam necessárias as contas referentes aos cortes que estivessem ativos na solução ótima. Por este motivo, alterou-se a programação para executar as contas apenas para estes cortes e não para todos.

Uma análise complementar descobriu que em dois pontos da rotina existia um conjunto de instruções que poderiam ser condensadas de forma a diminuir o consumo de tempo de processamento. Conforme está mostrado na Figura 41, os cálculos das parcelas da variável VAR estavam descriminados, de forma a tornar a programação mais clara, porém, esta forma de programar não é a mais eficiente, uma vez que existem cálculos que são desnecessariamente refeitos várias vezes.

```
VAR = A*B
VAR = VAR - (C-D)*E
VAR = VAR - (C-D)*F
IF( CONDICAO.EQ.1 ) THEN
   VAR = VAR + (C-D)*G
END IF
```

Figura 41 – Instruções na Forma Original

A condensação, com a correspondente simplificação, está apresentada na Figura 42, onde se pode notar a clara diminuição da quantidade de operações para se obter o mesmo resultado.

```
VAR = A*B – (C–D)*(E+F–G*CONDICAO)
```

Figura 42 – Instruções na Forma Modificada

Estas modificações foram implementadas e os novos resultados mostraram uma redução do tempo perfilável muito significativa, da ordem de quase 99%, conforme está mostrado na Tabela 5.

Tabela 5 – Diferenças de Tempo da Rotina que Calcula o Valor do π da Água

Rotina Calcula_Pi_Água	Tempo Médio (s)
Antes	192,89
Depois	2,17
Diferença (%)	-98,88

4.3.3.5 Análise da Rotina que Monta os Cortes de Benders

A análise dos tempos consumidos das linhas da rotina que monta os cortes de Benders indicou que um conjunto de linhas, que definiam se um corte seria inserido ou não, estavam com valores proporcionais bastante elevados. Observando a programação, percebeu-se que as seguintes modificações fariam com que os tempos consumidos diminuíssem:

- Teste condicional para decisão da inclusão do corte → a condição utilizada para decidir se um corte seria incluído no conjunto de cortes consistia de dois testes que uma determinada condição deveria atender, conforme pode ser visto na Figura 43.

```
IF( CONDICAO.GT.VALOR1 .AND. CONDICAO.GT.VALOR2 ) THEN

conjunto de tarefas

END IF
```

Figura 43 - Teste Condicional Original

Quando uma determinada condição tem que ser maior que dois valores, basta determinar qual dos dois valores é o maior e testar a condição se é maior que esse máximo, simplificando o teste, conforme está mostrado na Figura 44.

```
VALOR = MAX( VALOR1,VALOR2 )
IF( CONDICAO.GT.VALOR ) THEN

conjunto de tarefas

END IF
```

Figura 44 - Teste Condicional Modificado

- Saída do teste condicional para cortes idênticos → o corte é composto de valores em *n* dimensões e é necessário descobrir se ele é idêntico a um já anteriormente incluído. Logo, antes de inserir um corte é necessário verificar se já existe algum outro igual no conjunto de cortes. Este teste é feito através da comparação dos seus valores em todas as dimensões, bastando apenas um deles não atender determinadas condições para concluir que os dois cortes em comparação não sejam os mesmos. O problema é que o teste continuava a ser feito, mesmo depois de uma das dimensões ter atendido as duas condições. A modificação proposta é sair do teste assim que uma dimensão atender as condições, informando que o corte não é idêntico. Somente quando ocorrer semelhanças em dois cortes é que será necessária a verificação de todas as dimensões e isso ocorre poucas vezes ao longo do processo da aplicação.

As modificações propostas foram implementadas e os novos resultados mostraram uma redução do tempo perfilável bastante significativa, conforme está mostrado na Tabela 6.

Tabela 6 – Diferenças de Tempo da Rotina que Monta os Cortes de Benders

Rotina Montar_Cortes	Tempo Médio (s)
Antes	162,62
Depois	9,61
Diferença (%)	-94,09

Como a redução de tempo desta rotina impacta diretamente o desempenho da estratégia de paralelização, uma vez que ela define o conjunto de cortes que serão enviados para todos os processadores a partir do processador mestre, convém ressaltar o seguinte: apesar da redução de 94% no tempo ser um fato importante, que deve ajudar a melhorar o desempenho da estratégia de paralelização, ele não está levando em consideração os tempos gastos na leitura e escrita em disco que existem na rotina. Estas operações demoram muito mais no sistema operacional do que na instrução do

programa que dá estas ordens, não sendo contabilizadas no processo de perfilagem. Logo, a redução real do tempo gasto no processamento efetivo da rotina Montar_Cortes não será de 94%.

4.3.3.6 Análise da Rotina Altera o Vetor de Termos Independentes na Primeira Abertura da Primeira Série Hidrológica

A função desta rotina é montar o vetor de termos independentes na primeira abertura da primeira série hidrológica. Como neste ponto não existe ainda o problema montado na biblioteca de solução de PLs, a forma de acessar o vetor de os termos independentes difere do que é feito na rotina do item 4.3.3.1, que altera os valores diretamente na biblioteca. Porém, todos os procedimentos são similares, permitindo adotar as mesmas soluções que foram feitas na análise daquela rotina. Desta forma, as alterações foram as seguintes:

- Eliminação da inicialização do vetor auxiliar para montagem do vetor de termos independentes;
- Diminuição da quantidade de operações;
- Modificação de operação de teste condicional;
- Eliminação de operações intermediárias desnecessárias.

Com estas modificações implementadas na rotina, novos resultados de tempos de processamento foram obtidos através da perfilagem, indicando uma redução do tempo médio perfilável de aproximadamente 46,7%, conforme está mostrado na Tabela 7.

Tabela 7 – Diferenças de Tempo da Rotina para Alterar o Vetor de Termos Independentes na 1ª Abertura da 1ª Série

Rotina Alterar_Vetor_b_Inicial	Tempo Médio (s)
Antes	137,63
Depois	73,38
Diferença (%)	-46,68

4.3.3.7 Efeito de Todas as Alterações Simultaneamente

A execução da versão do programa com a programação de todas as modificações feitas pelo processo de perfilagem serial apresentou os resultados que estão mostrados na Tabela 8.

Tabela 8 – Tempos Médios Gastos no Processamento de Todas as Alterações da Perfilagem Serial

Rotinas	Tempo Médio (s)
Alterar_Vetor_b	749,82
Salvar_Solução	396,32
Montar_Vetor_b	344,27
Calcular_Pi_Água	2,31
Montar_Cortes	9,55
Alterar_Vetor_b_Inicial	73,51
Soma Tempos Rotinas	1.575,77
Tempo Total Perfilagem	1.712,92
Tempo Total Execução	88.733,00

Estes resultados mostram uma redução de 36,25% na soma dos tempos de perfilagem das 6 rotinas analisadas, uma vez que a soma destes tempos caiu de 2.471,61s para 1.575,77s, enquanto que o tempo total reduziu de 1,08%, por conta da redução de 89.702,67s (24h 55m 2s) para 88.733s (24h 38min 53s).

Com relação aos tempos perfiláveis de cada rotina, a comparação entre os tempos obtidos pelas versões com somente a alteração da rotina e com todas as alterações da perfilagem serial mostrou a existência de algumas variações para mais outras para menos, porém diante da grande dispersão possível no processo de perfilagem, todos os valores estão dentro da mesma ordem de grandeza. A maior variação negativa ocorreu na rotina que altera o vetor independente, onde o tempo caiu de 785,24s para 749,82 (-35,42s) e a maior variação positiva ocorreu na rotina que monta o vetor de termos independentes, onde o tempo consumido na perfilagem aumentou de 322,12s para 344,27s, ou seja, um acréscimo de 22,15s, apesar de não ter sido feita nenhuma alteração nesta rotina.

4.3.4 Desempenho da Estratégia de Paralelização Inicial

A primeira providência a ser feita para descobrir se será necessário melhorar o desempenho da estratégia de paralelização é executar o programa utilizando diferentes

quantidades de processadores e avaliar a evolução dos fatores de aceleração e das eficiências. Com este objetivo, executou-se o programa com diversas quantidades de processadores, cujos resultados estão apresentados na Tabela 9. Convém ressaltar que os resultados que serão apresentados são os tempos médios das simulações de cada caso. Caso seja necessária a consulta aos tempos individuais de alguma simulação, todos eles estão mostrados no Anexo A.

Tabela 9 – Resultados da Versão Inicial (Caso PMO Março 2009 – 200 Séries e 20 Aberturas)

Quantidade Processadores	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	88.317	-	-
2	45.245	1,95	97,60
4	23.351	3,78	94,55
8	13.045	6,77	84,63
16	7.781	11,35	70,94
32	5.896	14,98	46,81
40	4.945	17,86	44,65
64	4.495	19,65	30,70
80	3.841	22,99	28,74
128	3.362	26,27	20,52
200	2.712	32,56	16,28

Com os valores dos tempos definidos para cada quantidade de processadores, os valores dos fatores de aceleração e das eficiências foram calculados conforme as equações (45) e (46), respectivamente. Estes valores também estão apresentados de forma gráfica na Figura 45.

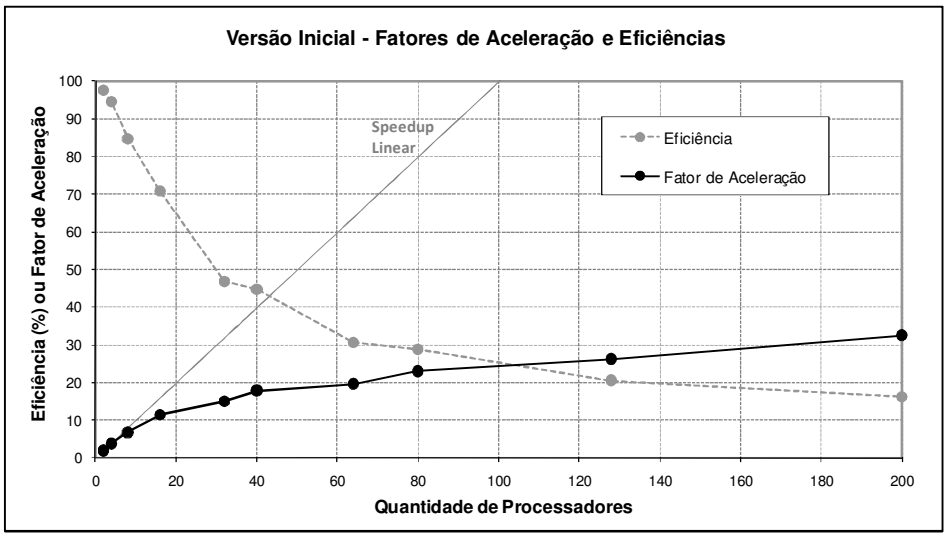

Figura 45 – Fatores de Aceleração e Eficiências da Estratégia de Paralelização Inicial

Pelos valores encontrados percebe-se que a estratégia de paralelização conseguiu reduzir o tempo de execução total de aproximadamente 24,5 horas para até 0,75 horas, ou seja, um fator de aceleração de quase 33 vezes. Não resta dúvidas que esta redução foi bastante significativa, porém a eficiência desta estratégia ainda apresenta valores baixos, principalmente nos casos a partir de 32 processadores.

A estratégia proposta apresentou elevadas eficiências para os casos com 2 e 4 processadores, 97,6% e 94,6%, respectivamente, reduzindo a eficiência para 84,6%, quando executada com 8 processadores, e para 70,9%, quando executada com 16 processadores. Estas quedas na eficiência sinalizam uma saturação no desempenho da estratégia de paralelização. Na execução dos casos com 16 processadores, além da saturação já verificada, a queda de desempenho também se deve à necessidade do uso de mais de uma placa *blade* na execução do processamento paralelo, já que em cada placa existem dois processadores com quatro núcleos de processamento cada, totalizando oito núcleos de processamento. O uso de mais de uma placa implica na utilização da rede de comunicação, que embora seja rápida, uma vez que a mesma possui velocidade de 1Gbps, ela é mais lenta do que a comunicação entre os processos dentro de uma mesma placa, gerando algum atraso para completar a sua execução.

Para quantidades a partir de 32 processadores, a eficiência não atingiu 50%, registrando valores muito baixos, de 20,5% e 16,3%, para os casos com 128 e 200 processadores, respectivamente, confirmando a característica de saturação conforme a quantidade de processadores vai aumentado.

Como o caso de PMO de 2009, utilizado como caso base, possui 200 séries hidrológicas e 20 aberturas nos ciclos *backward* e *forward*, casos com quantidade de processadores próxima deste valor (200) faz com que a distribuição de carga entre os processadores fique prejudicada. Para casos com poucos processadores, cada um recebe uma quantidade maior de problemas a resolver, desde modo, a possibilidade de existir PLs demorados e PLs de solução mais rápida aumenta, havendo certa compensação nos tempos finais do conjunto de problemas a serem resolvidos por cada um e, por conseqüência, fazendo com que a distribuição de carga seja mais uniforme. No caso limite, de 200 séries hidrológicas e 200 processadores, cada processador receberá uma quantidade pequena de problemas a resolver, aumentando muito o risco de existir uma distribuição de carga ruim.

Os resultados obtidos indicam que existe a necessidade de se identificar as razões que levaram à saturação da metodologia proposta com uma quantidade relativamente pequena de processadores. A saturação do processo pode ser causada por duas razões: (1) interna, que são os problemas de implementação e precisam ser corretamente identificados e corrigidos, tais como: pontos de sincronismos, balanceamento de carga deficiente, excesso de acesso a disco etc.; (2) externa, que pode ser a latência da rede de comunicação, problema com instruções de entrada e saída (I/O) etc. A eficiência da metodologia proposta de processamento paralelo deve ser melhorada através das modificações das razões internas, uma vez que estas estão ao alcance da correção. Já as razões externas são de modificações mais difíceis e não são alvos deste trabalho.

Capítulo 5.
Otimização da Estratégia de Paralelização

Neste capítulo estão apresentados os procedimentos utilizados para otimizar a estratégia de paralelização proposta no capítulo anterior. Esta otimização foi obtida a partir de modificações dos seguintes pontos da versão inicial:

- Diminuição da comunicação → obtida através da otimização do envio dos cortes;
- Balanceamento de carga → obtida com a criação de um processo auxiliar externo para gerenciar os problemas que os processos principais devem resolver;
- Introdução de assincronismo → obtida com a modificação da forma como os cortes de Benders são agrupados;
- Adequação à arquitetura de *hardware* → obtida com a criação de novos comunicadores para que as comunicações entre os núcleos de processamento de um mesmo nó, mais rápidas, possam ser utilizadas, diminuindo o tempo de comunicação.

É importante destacar que os pontos relacionados acima são etapas que devem ser sempre avaliadas na otimização de qualquer estratégia de paralelização, uma vez que a sua modificação pode reduzir significativamente o tempo final de execução.

5.1 Versão 2 – Otimização do Envio dos Cortes

Para se iniciar o processo de perfilagem paralela do programa, decidiu-se por executar a estratégia de paralelização com 32 processadores. A escolha desta quantidade se deve aos seguintes fatores: (1) esta quantidade é bastante utilizada nas execuções do programa; (2) o tempo final do caso executado não é longo, de forma a não atrasar a

obtenção de resultados e as respectivas análises; (3) com esta quantidade de processadores, o desempenho do programa já está bastante deteriorado.

A maior parte do processamento ocorre na solução dos ciclos *backward* e *forward*, realizados nas rotinas *backward* e *forward*, respectivamente, e poder-se-ia incluir as instruções de tomada de tempo nas duas rotinas. Porém, como pode ser visto na Tabela 10, o tempo de processamento gasto na solução do ciclo *backward* é muito superior ao que é gasto na solução do ciclo *forward*. Por esta tabela, percebe-se que, em média, se gasta aproximadamente 5,3 vezes mais tempo de processamento no ciclo *backward* do que no *forward*. Logo, num primeiro instante, será dada uma atenção apenas aos tempos de processamento do ciclo *backward*, pois um melhor desempenho do programa neste ciclo terá um impacto muito maior no tempo final do programa do que o do ciclo *forward*.

Tabela 10 – Relação dos Tempos de Execução dos Ciclos *Backward* e *Forward*

Iter.	Ciclo *Backward*				Ciclo *Forward*				Relação Back/Forw
	h	m	s	s	h	m	s	s	
1	0	1	7	67	0	0	7	7	9,57
2	0	1	35	95	0	0	19	19	5,00
3	0	2	12	132	0	0	27	27	4,89
4	0	2	56	176	0	0	46	46	3,83
5	0	3	28	208	0	0	45	45	4,62
6	0	4	13	253	0	0	53	53	4,77
7	0	4	43	283	0	1	4	64	4,42
8	0	5	21	321	0	1	8	68	4,72
9	0	5	54	354	0	1	11	71	4,99
10	0	6	29	389	0	1	16	76	5,12
11	0	7	8	428	0	1	21	81	5,28
12	0	7	41	461	0	1	27	87	5,30
13	0	8	20	500	0	1	29	89	5,62
14	0	8	53	533	0	1	37	97	5,49
15	0	9	36	576	0	1	41	101	5,70
Média									5,29

- **Identificação e Diagnóstico do Problema**

A escolha dos pontos de tomada de tempo se baseou em tarefas macros executadas na rotina, destacando blocos de leitura de dados, envio de mensagens, processamento dos PLs etc. Estes pontos de medição e os respectivos tempos gastos na execução das tarefas estão disponíveis no item B.1 do Anexo B.

Dos tempos obtidos, o mais significativo foi o referente à montagem e solução dos PLs, armazenamento da solução ótima e cálculos para gerar os cortes de Benders,

seguido pelo ponto referente ao envio dos cortes do processador mestre para os demais. Como o primeiro ponto se refere a uma série de tarefas e também à solução dos PLs, era de se esperar que o tempo consumido neste ponto fosse realmente o maior. O segundo colocado foi o envio dos cortes para os outros processadores, significando que este envio está tomando um tempo de processamento significativo, uma vez que, por exemplo, na 15ª iteração o tempo gasto com esta comunicação foi de aproximadamente 20% (114,94/582,61 - Tabela 48 – Anexo B) do tempo total de processamento do processador mestre.

Com relação aos demais processadores, o maior tempo consumido também foi no ponto que inclui, dentre outras tarefas, a montagem e solução dos PLs, seguido pelo recebimento dos cortes do processador mestre. Porém, o tempo médio consumido no recebimento foi muito mais significativo do que o ocorrido com o envio no processador mestre, já que este em média consumiu aproximadamente 40% (237,82/582,33 - Tabela 48 – Anexo B) do tempo da última iteração. Para efeito de comparação, o tempo médio de processamento da solução dos PLs, que é a principal tarefa do ponto onde é esperado o maior consumo do tempo de processamento, foi aproximadamente de 55% do tempo total desta iteração ((272,79+49,66)/582,33 - Tabela 48 – Anexo B).

A análise, principalmente dos resultados médios dos demais processadores, indica que a diminuição do desempenho do programa está ligada fortemente a um problema de comunicação entre o processador mestre e os demais processadores no envio/recebimento dos cortes.

- **Solução Proposta**

Como o tamanho de memória necessário para o armazenamento dos cortes seria muito extenso, o programa armazena na memória RAM somente os cortes que serão utilizados nos PLs do período que está sendo resolvido. Todos os outros cortes são armazenados em disco e apenas no processador mestre. Outra informação importante é que a quantidade de cortes aumenta com o aumento das iterações, podendo ser adicionados, para um caso de PMO, até 200 novos cortes a cada iteração. Para um caso que convergisse com 15 iterações, é possível ter até 3.000 cortes nos PLs da 15ª iteração, porém este número é um pouco menor devido à eliminação de cortes redundantes, principalmente nas primeiras iterações. A estratégia implementada na

proposta inicial para o tratamento e envio dos cortes pode ser visualizada na Figura 46, mostrada a seguir.

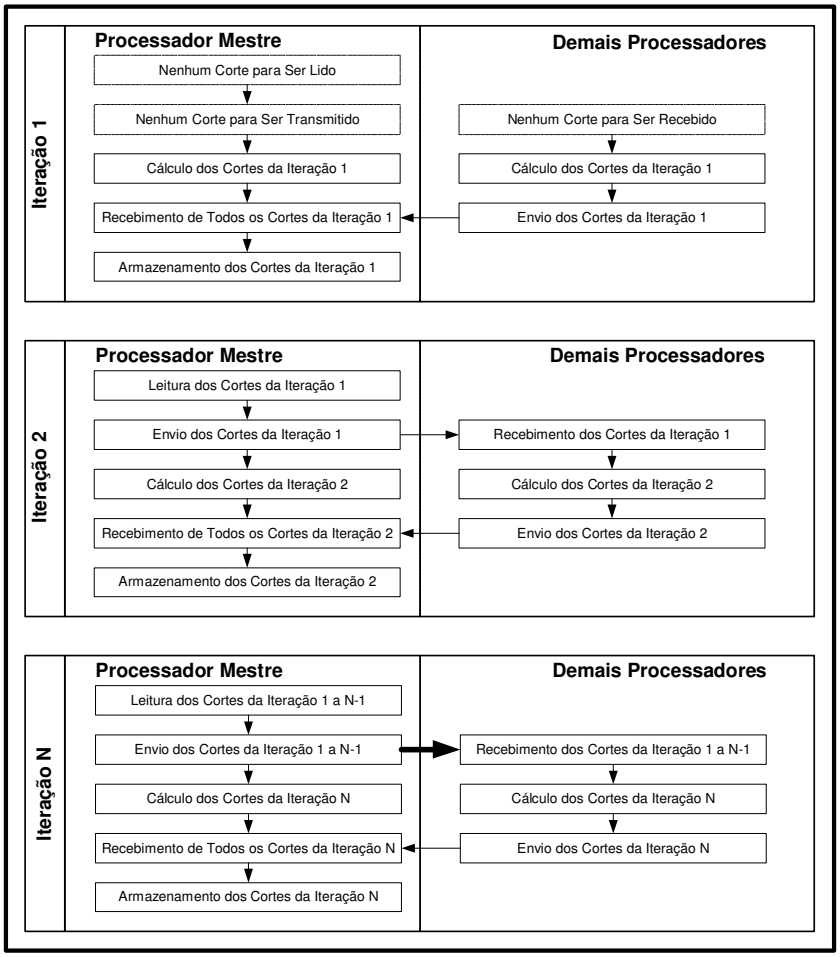

Figura 46 - Estratégia Original para Armazenamento e Envio dos Cortes

Pode-se observar que conforme o número de iterações vai aumentando, a quantidade de cortes a ser transmitida do processador mestre para os demais vai aumentando, visto que os cortes utilizados nas iterações anteriores não ficam armazenados na memória nem num disco local. Logo, esta grande quantidade de cortes é transmitida a cada período de cada iteração do processador mestre para todos os outros processadores. Esta é a razão do grande tempo de processamento gasto no envio e recebimento dos cortes pelos processadores. Um ponto importante a destacar é que os cortes de uma determinada iteração são compostos pelos cortes gerados na iteração propriamente dita e também por aqueles gerados nas iterações anteriores. Ou seja,

enquanto um determinado corte não for descartado, ele é transmitido pelo processador mestre em todas as iterações subseqüentes àquela em que ele foi gerado.

A solução para este problema é reduzir a quantidade de cortes a serem transmitidos, armazenando localmente os cortes já recebidos anteriormente. Ou seja, será feita a substituição de comunicação de rede por armazenamento em disco. Nesta nova estratégia, mostrada na Figura 47, os demais processadores passam a armazenar os cortes que foram recebidos do processador mestre. Com isso, não é mais necessário transmitir todos os cortes que forem lidos no processador mestre, bastando enviar apenas os cortes que foram gerados na iteração imediatamente anterior à iteração corrente. O ponto escolhido para o armazenamento dos cortes nos demais processadores é o mesmo do cálculo e armazenamento dos cortes feito pelo processador mestre. Ou seja, enquanto o processador mestre estiver calculando e armazenando os cortes referentes à iteração corrente e que serão utilizados na próxima iteração, os demais processadores, que estariam ociosos, passam a armazenar os cortes que foram recebidos no início do período.

Já o ponto escolhido para a leitura dos cortes nos demais processadores coincide com o ponto de leitura dos cortes no processador mestre, visto que neste ponto os demais processadores estariam ociosos a espera dos cortes que seriam enviados pelo processador mestre.

O impacto desta nova estratégia é nulo na primeira iteração e pequeno nas iterações iniciais porque a diferença de quantidade de cortes a serem transmitidos do processador mestre entre as duas estratégias é pequena. O ganho de tempo será mais significativo com o avanço do número de iterações, visto que em vez de se transmitirem todos os cortes, apenas os da iteração imediatamente anterior serão transmitidos. Também é esperado que a nova estratégia tenha um desempenho melhor em casos com quantidades maiores de processadores por conta da diminuição dos tamanhos das mensagens a serem comunicadas entre eles.

Uma conseqüência bastante interessante desta nova estratégia ocorre na rotina *forward*, uma vez que o envio dos cortes não é mais necessário porque todos eles já estarão armazenados localmente para todos os processadores, logo, o tempo de processamento gasto nesta rotina também deverá diminuir, pois este ponto de transmissão de dados foi eliminado.

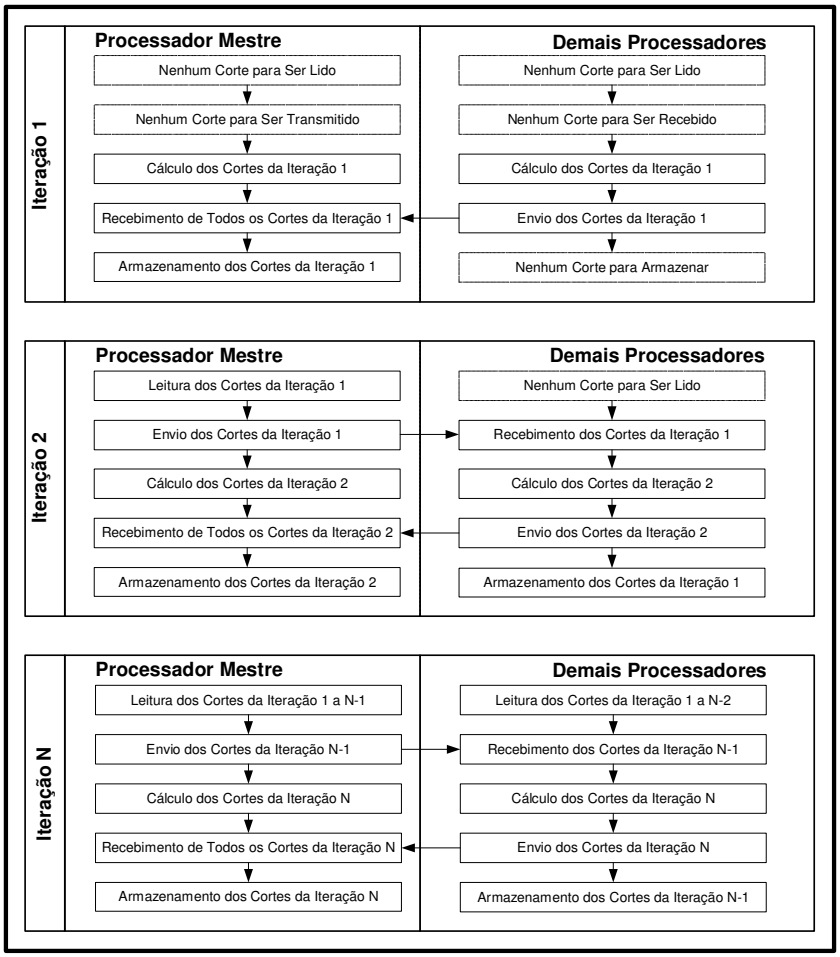

Figura 47 - Nova Estratégia para Armazenamento e Envio dos Cortes

- **Resultados da Solução Proposta**

Comparando os resultados do processador mestre desta nova versão, que também estão disponíveis no Anexo B, com os da versão inicial, percebe-se claramente a redução quase integral do tempo de envio dos cortes, mostrando uma diminuição bastante significativa no tempo que o processador mestre gastava com o envio dos cortes para os demais processadores. O tempo total gasto na 15ª iteração, que era de 582,61s, com 114,94s consumido no envio dos cortes (Tabela 48 – Anexo B), passou a ser de 460,17s, com 0,52s consumido no envio dos cortes (Tabela 50 – Anexo B). A redução de tempo total foi de 21%, sendo que o tempo ganho (122,44s) foi praticamente todo ele devido à mudança da estratégia de tratamento dos cortes.

Na comparação com os tempos médios dos demais processadores da versão inicial, os dois pontos com os maiores gastos de tempo de processamento foram os mesmos, porém, o tempo de processamento médio gasto no recebimento dos cortes foi menor do que o ocorrido na versão inicial. Nesta versão, o tempo de processamento médio das tarefas referentes ao recebimento dos cortes, na 15ª iteração, foi de 119,14s (Tabela 50 – Anexo B), ante os 237,82s (Tabela 48 – Anexo B) da versão inicial, ou seja, uma redução de 118,68s. Esta diferença de tempo se refletiu no ganho final médio da 15ª iteração, que foi de 123,42s, uma vez que a nova versão executou a última iteração em 458,91s (Tabela 50 – Anexo B) e a versão original executou esta iteração em 582,33s (Tabela 48 – Anexo B), ou seja, uma redução de tempo de aproximadamente 21,2%.

Para avaliar esta nova versão, nomeada de versão 2, novamente vários casos foram executados com diversas quantidades de processadores, conforme estão apresentados na Tabela 11.

Tabela 11 – Resultados da Versão 2

Quantidade Processadores	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	88.136	-	-
2	45.182	1,95	97,53
4	23.334	3,78	94,43
8	12.972	6,79	84,93
16	7.218	12,21	76,32
32	4.231	20,83	65,10
40	3.510	25,11	62,77
64	2.748	32,08	50,12
80	2.482	35,51	44,38
128	1.911	46,13	36,04
200	1.605	54,90	27,45

Os valores dos fatores de aceleração e das eficiências estão mostrados graficamente na Figura 48. Com relação à versão inicial, percebe-se um ganho significativo para os casos com maiores quantidades de processadores. Uma comparação completa entre as duas versões está mostrada na Tabela 12.

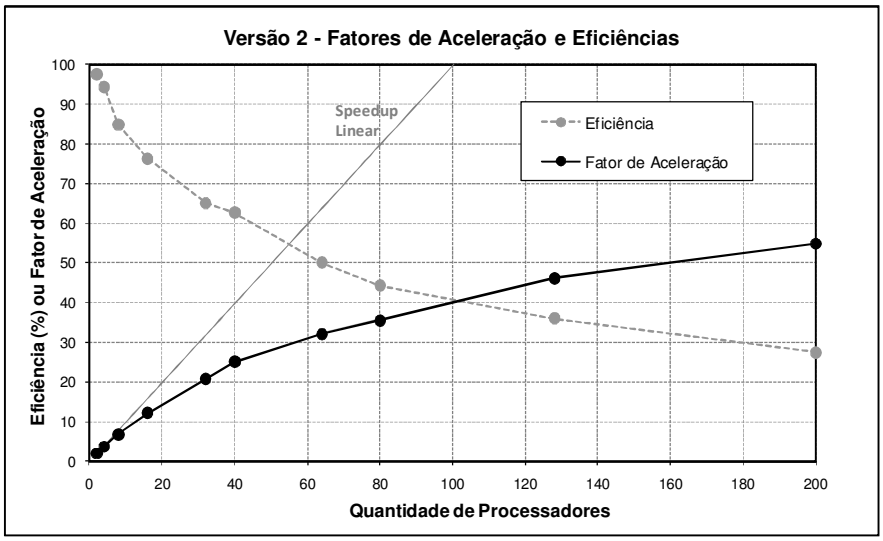

Figura 48 - Fatores de Aceleração e Eficiências da Versão 2

Na comparação entre as duas versões, fica confirmada uma redução de tempo a partir do caso com 16 processadores, tornando bastante significativa a partir do caso com 32 processadores, atingindo a redução máxima de 43,16% no caso com 128 processadores.

Tabela 12 – Comparação dos Tempos Médios entre as Versões Inicial e a 2

Quantidade Processadores	Versão Inicial (s)	Versão 2 (s)	Diferença de Tempos (%)
1	88.317	88.136	-0,21
2	45.245	45.182	-0,14
4	23.351	23.334	-0,07
8	13.045	12.972	-0,56
16	7.781	7.218	-7,23
32	5.896	4.231	-28,24
40	4.945	3.510	-29,02
64	4.495	2.748	-38,87
80	3.841	2.482	-35,38
128	3.362	1.911	-43,16
200	2.712	1.605	-40,81

5.2 Versão 3 - Balanceamento Dinâmico de Carga dos Processadores

Ao longo da análise dos resultados da versão 2, observou-se uma diferença significativa entre os somatórios dos tempos de processamento referentes aos PLs das demais aberturas da primeira série hidrológica e aos PLs das aberturas das demais séries hidrológicas. Apesar de nestes pontos ocorrerem a execução de diversas tarefas, o que pode justificar as diferenças de tempos de processamento são as soluções dos PLs

existentes nestes dois pontos de tomada de tempo. Logo, esta diferença significativa entre os tempos consumidos nos processadores implica num balanceamento deficiente na estratégia de paralelização da versão 2. Esta terceira versão consiste numa solução para este problema.

- **Identificação e Diagnóstico do Problema**

Apesar do significativo ganho de tempo conseguido com a versão anterior, observou-se que os tempos das soluções dos PLs dos processadores estavam significativamente heterogêneos, como pode ser comprovado pelos tempos mostrados na Tabela 51, disponibilizada no item B.2 do Anexo B, correspondente à 15ª Iteração do ciclo *backward*.

Estas diferenças se devem principalmente ao processo atual da divisão dos problemas pelos processadores. Na versão atual da estratégia de paralelização, a distribuição dos problemas para cada processador é feita através da divisão da quantidade de séries hidrológicas pela quantidade de processadores. Caso esta operação resulte em resto, um grupo de processadores recebe um problema a mais até que todas as séries estejam distribuídas para algum processador. Por exemplo, num caso com 200 séries hidrológicas e 32 processadores, a divisão de 200 por 32 resulta em 6,25, fazendo com que 24 processadores, 3/4 do total, recebam 6 séries e 8, 1/4 do total, recebam 7 séries. Convém ressaltar que o recebimento de uma determinada série hidrológica significa a solução dos PLs referentes às aberturas com diferentes cenários de afluência daquela série no ciclo *backward*. A probabilidade desta distribuição de carga estática ser eficiente é muito pequena e é necessário programar um processo dinâmico para distribuir as séries hidrológicas entre os processadores, minimizando o tempo ocioso e, conseqüentemente, melhorando o desempenho da estratégia de paralelização.

- **Solução Proposta**

A solução para o problema da distribuição dinâmica dos problemas para os processadores passa pela criação de um gerenciador de solução dos PLs. Este gerenciamento pode ser feito deslocando-se um dos processadores para esta função. A vantagem desta solução é que o processo responsável pelo gerenciamento não concorre

com nenhum dos processos que estão executando a solução dos PLs. A desvantagem obvia é a perda de um dos processadores na solução dos PLs, sobrecarregando de alguma forma os demais.

Outra possível forma de programar o gerenciamento da solução dos PLs é a criação de processo externo à aplicação. Neste caso, o processo externo será executado num determinado processador, concorrendo com este durante a solução dos PLs. O que se espera é que esta concorrência não seja muito significativa, uma vez que o processo externo somente receberá e enviará pequenas mensagens para os processadores que estão executando a solução dos PLs. Além disto, a pequena perda de desempenho do processador que executar o processo externo será compensada com a distribuição automática das séries hidrológicas para outros processadores. Como pode ser percebida, a vantagem desta solução é que não se "perde" nenhum processador para executar a solução dos PLs.

O gerenciamento externo é feito através de um programa, desenvolvido em Fortran, que é disparado através da instrução MPI_COMM_SPAWN, que precisa ser executada por todos os processadores que participam da execução do processo original, pois esta é uma instrução coletiva, conforme está representado na Figura 49. A execução desta instrução faz com que seja criado um intercomunicador, chamado de INTERCOMM, entre os processos que realizam o processamento principal (processos pais) e o processo de gerenciamento da solução dos PLs (processo filho), permitindo a troca de mensagens entre eles.

Como o processo de gerenciamento da solução dos PLs não participa diretamente do processamento principal, é necessário enviar algumas variáveis de controle do caso que está sendo executado. Esta comunicação é feita através do processador mestre dos processos pais, conforme está mostrado na Figura 50.

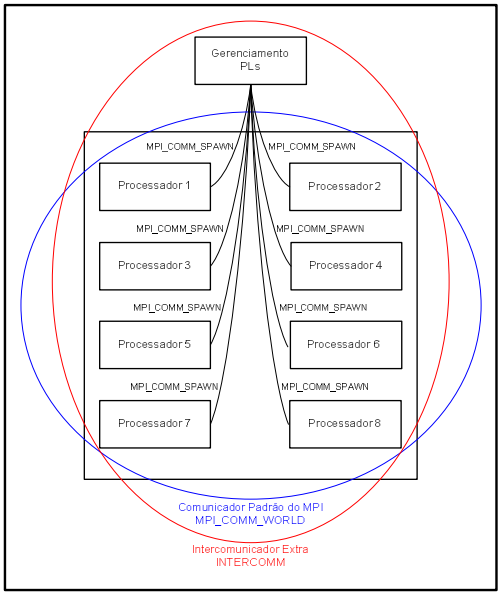

Figura 49 - Criação do Processo de Gerenciamento Externo para a Solução dos PLs (processo filho)

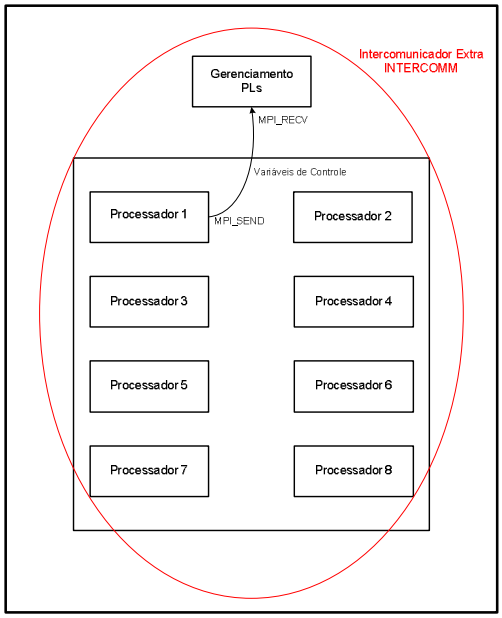

Figura 50 – Envio de Variáveis de Controle para o Processo Filho

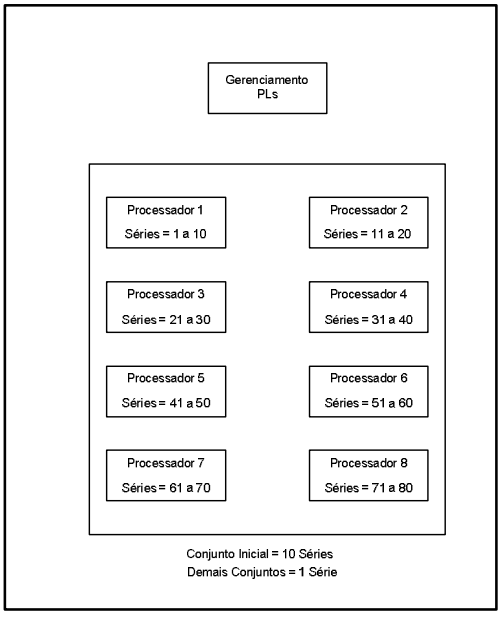

Figura 51 – Distribuição do Conjunto Inicial das Séries Hidrológicas para Cada Processador

Além destas mensagens, o processo filho também recebe duas outras variáveis, que devem ser informados num novo arquivo de dados do processo pai. Estes valores correspondem aos tamanhos do conjunto inicial e dos conjuntos subseqüentes de séries hidrológicas. O conjunto inicial corresponde à quantidade de séries que cada processo receberá inicialmente. Nesta etapa, não é necessária nenhuma comunicação entre os

processos pais e entre estes e o processo filho. A distribuição das séries do conjunto inicial é dependente apenas da identificação de cada processador, ou seja, o processador mestre possui a informação do seu conjunto de séries iniciais, o segundo processador também possui a informação do seu conjunto e assim por diante, conforme pode ser visto na Figura 51. Nesta figura é dado um exemplo de como seria a distribuição do conjunto inicial das séries hidrológicas de cada processo, se o tamanho do conjunto inicial fosse composto de 10 séries e os conjuntos subseqüentes com apenas 1 série. É importante ressaltar que, no caso da rotina *backward*, quando um processador recebe uma determinada série, ele fica responsável para resolver todos os problemas relacionados com as aberturas dos cenários de afluências daquela série, que, num caso de PMO, corresponde a 20 PLs.

Os tamanhos dos conjuntos iniciais e subseqüente são de livre escolha, permitindo que estes valores possam ser ajustados para cada tipo de computador em que a aplicação seja executada. A escolha de conjuntos pequenos faz com que a quantidade de comunicação entre os processos pais e o processo filho aumente, porém a carga de trabalho dos processos pais tende a ficar mais homogênea, melhorando o desempenho da execução em paralelo. Já a escolha de tamanhos maiores para os conjuntos de séries diminui o tempo destinado à comunicação entre os processos, porém a carga de trabalho dos processadores tende a ficar desequilibrada, podendo piorar o desempenho global do processamento paralelo. Logo, existe uma relação de compromisso entre ganhar tempo diminuindo a comunicação e perder desempenho por conta do risco do aumento do desequilíbrio da carga nos processadores. O ajuste destes valores é feito de forma empírica e dependerá do tipo de configuração dos componentes de cada computador.

Depois que um determinado processador resolve todos os seus problemas, ele envia um mensagem para o processo de gerenciamento de PLs, que envia uma mensagem de volta com o intervalo de séries a serem resolvidas. Este intervalo de séries dependerá do tamanho dos conjuntos subseqüentes. Para o caso em que o tamanho do conjunto seja 1, os valores inicial e final do conjunto de séries serão iguais. Um exemplo desta troca de informações pode ser visualizado na Figura 52 e na Figura 53.

Neste exemplo o processador 6 termina a solução dos PLs destinados a ele e envia uma mensagem para o processo de gerenciamento de PLs, que está em modo de espera até a chegada da mensagem. Ao receber a mensagem, o processo de

gerenciamento de PLs identifica o remetente da mesma e retorna para ele o intervalo das séries a serem resolvidas, como neste caso o tamanho é unitário, os dois valores serão iguais. No exemplo da Figura 53, o processador 6 recebe a informação que deverá resolver os problemas relacionados com a série hidrológica 81. Como todos os dados para a solução de qualquer série já estão disponíveis para todos os processadores, basta que ele receba a identificação de qual(is) série(s) resolver para montar e executar os problemas.

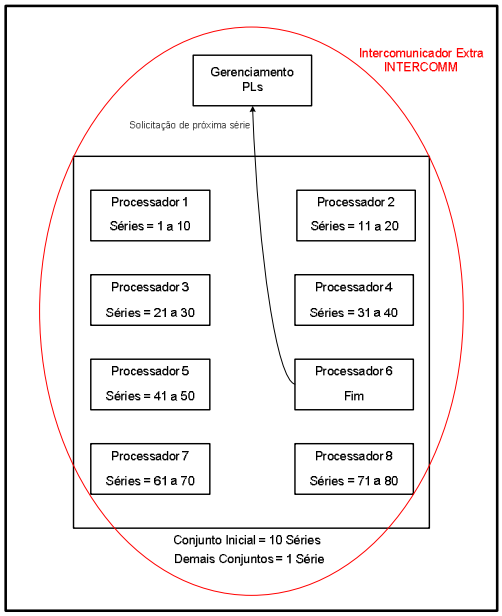

Figura 52 – Informação de Final de Processamento de Séries do Processador 6

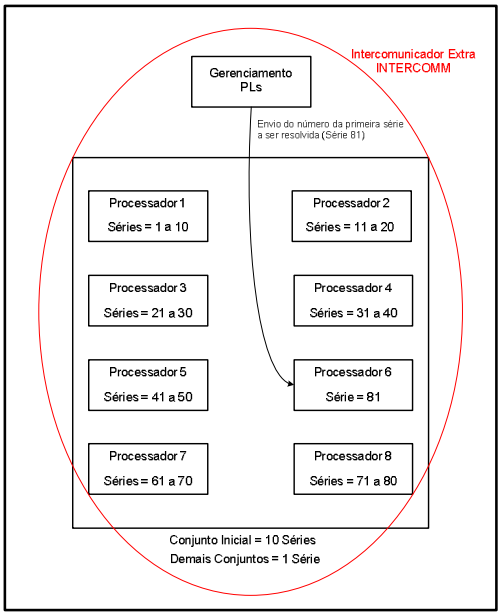

Figura 53 – Envio do Próximo Conjunto de Séries para o Processador 6

Este procedimento é realizado por todos os processadores conforme vai terminando a solução dos PLs sob sua responsabilidade. O processo de gerenciamento de PLs vai enviando os conjuntos a cada solicitação até terminar a quantidade total de séries a serem simuladas, quando então passa a enviar o valor "-1" em vez do valor das séries. No final, todos os processadores receberão este valor, informando que não existem mais séries a serem processadas.

- **Resultados da Solução Proposta**

Da mesma forma como foi feito com a versão 2, para avaliar esta nova versão vários casos foram executados com diversas quantidades de processadores e os seus valores médios estão apresentados na Tabela 13.

Tabela 13 - Resultados da Versão 3

Quantidade Processadores	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	88.173	-	-
2	44.389	1,99	99,32
4	22.496	3,92	97,99
8	12.417	7,10	88,76
16	6.648	13,26	82,90
32	3.826	23,05	72,02
40	3.298	26,73	66,83
64	2.410	36,59	57,17
80	2.236	39,43	49,29
128	1.756	50,21	39,23
200	1.633	53,98	26,99

Os valores dos fatores de aceleração e das eficiências estão mostrados graficamente na Figura 54, onde se percebe que o fator de aceleração das simulações até 16 processadores estão bem próximos da reta de *speedup linear*.

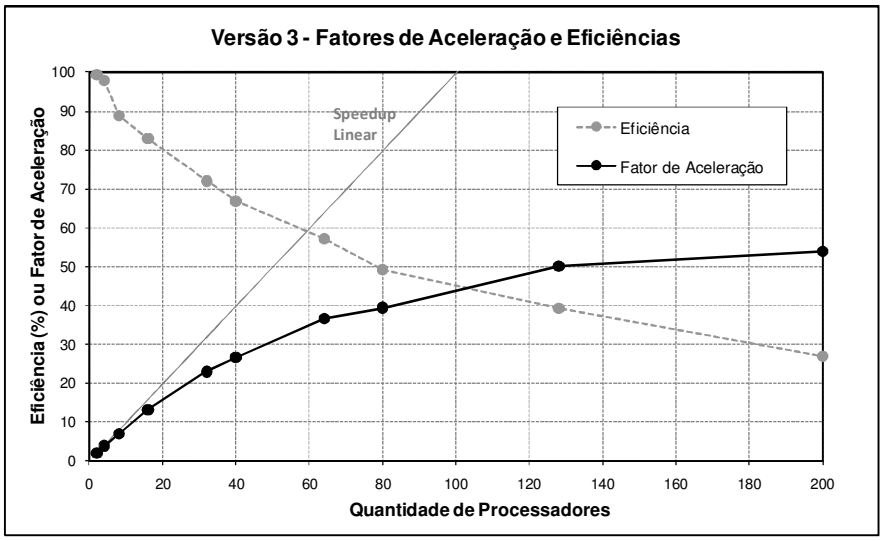

Figura 54 - Fatores de Aceleração e Eficiências da Versão 3

Para comparar estes resultados com os da versão anterior, a Tabela 14 mostra os tempos médios das execuções com as diferentes quantidades de processadores e as respectivas diferenças percentuais de tempo. A primeira constatação destes resultados é que houve um ganho de eficiência em quase todas as quantidades de processadores, exceção apenas no caso de 200 processadores. Esta conclusão se deve ao fato dos tempos médios para 1 processador não variaram, enquanto que houve queda nos outros tempos médios. Outra constatação foi que os ganhos foram menores para as quantidades

menores de processadores, porém estes casos já possuíam altas eficiências na versão 2 e a margem para melhorá-las era pequena. O caso com 2 processadores possuía uma eficiência de 97,53% e a mesma aumentou para 99,32%, apresentando uma redução de tempo de 1,76%. Já o caso com 4 processadores também passou a ter uma eficiência bastante elevada de 97,99%, com uma redução de 3,59% no tempo médio de processamento. Os casos com as demais quantidades de processamento, exceto o de 200 processadores, que possuíam menores eficiências, o gerenciamento externo na solução dos PLs fez com que a redução de tempo ficasse entre 4,28% e 12,29%.

Tabela 14 - Comparação dos Tempos Médios entre as Versões 2 e 3

Quantidade Processadores	Versão 2 (s)	Versão 3 (s)	Diferença de Tempos (%)
1	88.136	88.173	0,04
2	45.182	44.389	-1,76
4	23.334	22.496	-3,59
8	12.972	12.417	-4,28
16	7.218	6.648	-7,90
32	4.231	3.826	-9,58
40	3.510	3.298	-6,03
64	2.748	2.410	-12,29
80	2.482	2.236	-9,92
128	1.911	1.756	-8,09
200	1.605	1.633	1,74

O perfil de ganho de tempo foi aumentando com o aumento da quantidade de processadores utilizada até atingir o máximo na simulação com 64 processadores. O ganho da simulação com 40 processadores foi um pouco menor por conta da particularidade desta quantidade de processadores, que gera uma divisão perfeita para o caso com 200 séries hidrológicas e a distribuição estática anteriormente adotada. Esta divisão sem restos fez com que a distribuição dos problemas fosse homogênea, pelo menos, em quantidade, dando uma tendência de melhora à eficiência deste caso na versão com distribuição de carga estática.

A simulação com 200 processadores apresentou um aumento de tempo em vez de redução, mas este fato é facilmente explicável: um caso de PMO possui 200 séries e quando um caso com 200 processadores é executado, a distribuição de uma série hidrológica para cada processador é feita e o gerenciamento externo não tem nenhum efeito, a não ser piorar um pouco o caso, pois nesta versão cada processador precisa receber um sinalizador extra, indicando que não existe mais nenhuma série a resolver.

5.3 Versão 4 - Otimização do Agrupamento dos Cortes

Durante a perfilagem da versão 3, observou-se que o agrupamento dos cortes, que é executado pelo processador mestre após a solução de todos os PLs, poderia ser executado conforme os PLs fossem resolvidos, de forma a diminuir a ociosidade dos demais processadores durante a realização do agrupamento propriamente dito. Logo, esta quarta versão diz respeito à identificação, solução e aos resultados da mudança nesta forma de agrupar os cortes.

- **Identificação e Diagnóstico do Problema**

A implementação original do agrupamento dos cortes, ainda remanescente da programação serial original, previa que o mesmo seria executado após as soluções de todos os PLs, uma vez que o processador mestre, que participava normalmente da solução dos PLs, estaria disponível para receber os cortes médios gerados por todos os processadores. Uma vez recebidos estes cortes médios, o agrupamento era feito seguindo sempre a mesma seqüência das séries hidrológicas, de forma a evitar que ao se modificar a quantidade de processadores, resultados pudessem ser diferentes por conta da existência de PLs com múltiplas soluções. Esta implementação pode ser observada na Figura 55, a partir de um exemplo com 32 problemas e 16 processadores.

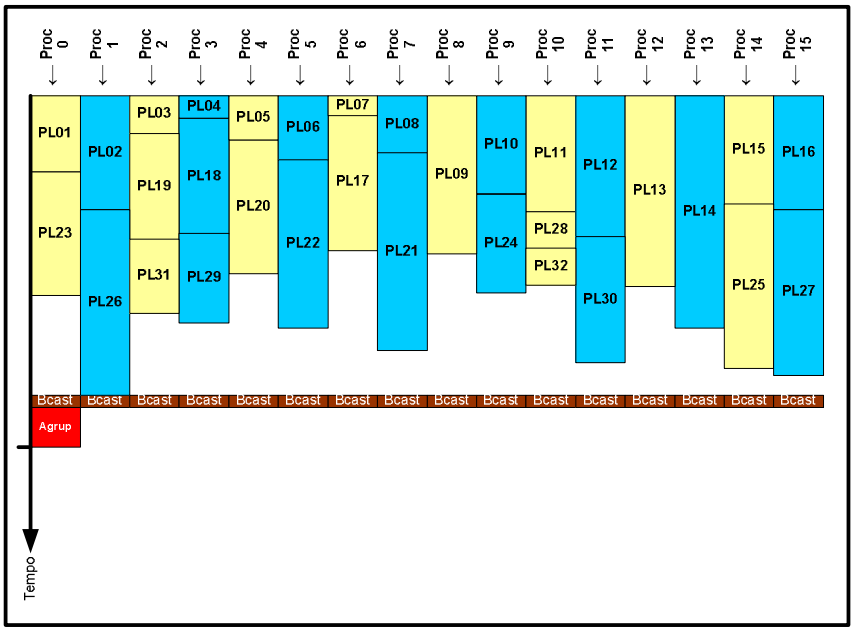

Figura 55 – Agrupamento Anterior dos Cortes

Pode-se observar que a distribuição dinâmica dos PLs faz com que a quantidade de problemas que cada processador resolva seja variável, de forma a tentar equalizar o tempo total gasto por cada um. Por conta disto, no exemplo mostrado, alguns processadores resolveram apenas um problema, enquanto que outros conseguiram resolver três problemas. No exemplo, tem-se 16 processadores e 32 problemas a serem resolvidos, logo, inicialmente, cada processador recebe um problema para resolver, com isso os 16 primeiros problemas são distribuídos para os 16 processadores, resultando em um problema por processador. Assim que o problema é finalizado, o processador recebe o primeiro problema disponível para ser resolvido. No exemplo da Figura 55, o primeiro problema resolvido é o PL07, por conta disto, o processador 6 recebe o PL17 como segundo problema a resolver. Ao final da solução dos 32 PLs, todos os processadores executam uma instrução coletiva de *broadcast*, de forma a disponibilizar todos os resultados para o processador 0, que executa o agrupamento dos cortes, eliminando os que forem redundantes e armazenando os que forem efetivos. Esta tarefa é executada pelo processador 0 após o recebimento de todos os resultados. Esta implementação possui o inconveniente de manter vários processadores ociosos durante o tempo entre o fim da solução dos seus últimos problemas e a instrução para envio dos resultados para o processador 0, que é realizada após o final da solução do último problema.

- **Solução Proposta**

Apesar do gerenciamento dinâmico já ter sido implementado com sucesso, tendo melhorado a eficiência da estratégia de paralelização, diminuindo as diferenças de tempo de processamento gasto por cada processador, o processamento não é igualitário, principalmente para os casos em que a quantidade de processadores for similar à quantidade de séries hidráulicas que serão distribuídas. Num caso limite, como por exemplo, um caso PMO com 200 séries hidráulicas e 200 processadores, não há como gerenciar a quantidade de séries por processador uma vez que todos receberão a primeira série e não haverá mais nenhuma a ser distribuída.

Para os casos em que a quantidade de séries for muito maior que a quantidade de processadores, a diferença dos tempos de processamento tende a ser menor por conta do gerenciamento dos PLs, logo, o tempo ocioso dos processadores tende a ser pequeno.

Uma maneira de diminuir o tempo ocioso dos processadores durante a etapa de agrupamento dos cortes seria o envio dos mesmos assim que fossem obtidos, de forma que o agrupamento começaria muito antes do último corte chegar. Neste caso, quando chegasse o último corte, seria esperado que poucos cortes ainda faltassem a serem agrupados, resultando numa redução do tempo após a solução da última série, bem diferente do que é feito atualmente.

Na nova implementação proposta, como pode ser visualizada na Figura 56, o processador 0 é retirado do conjunto de processadores que resolvem os PLs que geram os cortes. Isto foi necessário para que o processador responsável pelo agrupamento dos cortes estivesse sempre disponível para receber as mensagens dos outros processadores. Caso este processador estivesse comprometido com a solução dos PLs, haveria um risco elevado de acontecer um retardamento no recebimento dos cortes, por conta da realização das duas tarefas simultaneamente. Evidentemente que a retirada de um processador da solução dos problemas acarreta num aumento de carga para os demais processadores, podendo resultar num tempo maior para a solução de todos os problemas. Este fato pode ser minorado quando a quantidade de processadores do caso for de tal monta que a retirada de um não seja significativa na solução total do caso. Para um caso com 2 ou 4, não é recomendável retirar um processador da solução dos PLs. Para os casos de 8 e 16, é necessário avaliar a situação com a execução de casos e verificação dos tempos totais de processamento. Para quantidades a partir de 32 processadores, é recomendável que a nova implementação seja adotada.

Nesta nova forma de agrupar os cortes os problemas de 1 a 15 são distribuídos para os processadores de 1 a 15. Assim que o problema é finalizado, o respectivo processador envia os cortes para o processador 0 armazená-los. No caso deste exemplo, o primeiro processador que termina o seu problema é o sete, que envia imediatamente os resultados, representado por uma faixa marrom na figura. Neste ponto ocorre uma restrição: assim que o processador 0 recebe os cortes, ele já poderia começar a etapa de agrupamento deste corte, porém, não há garantia de que estes cortes cheguem sempre na mesma ordem. Como uma simples mudança na ordem de execução dos cortes no agrupamento pode mudar os resultados em PLs com múltiplas soluções, deve-se estabelecer uma ordem sempre fixa para evitar tal fato. Adotou-se a mesma ordem que existia na implementação original, ou seja, o agrupamento será iniciado pelo corte

gerado pela série um, depois pela série dois, até chegar a última série. Como não há garantia de que os resultados chegarão nesta seqüência adotou-se o seguinte procedimento: quando o processador 0 recebe alguma mensagem, a primeira posição do vetor de resultados é a posição da série, que é armazenada e marcada como já recebida. Depois disto, ocorre a verificação de qual é a próxima série a ser agrupada. Caso os resultados que acabaram de chegar não sejam da próxima série a ser agrupada, eles somente são armazenados e o processamento volta para receber a próxima mensagem. Caso a série recém chegada seja a próxima série a ser agrupada, após o seu armazenamento ela é enviada para a rotina que executa o agrupamento. Neste ponto, o processador 0 verifica se a próxima série a ser agrupada também já foi recebida anteriormente, se afirmativo, ela também é agrupada, assim como todas as seguintes que já foram recebidas. Após o agrupamento da última série da seqüência, o processamento volta à atenção para o recebimento da próxima mensagem.

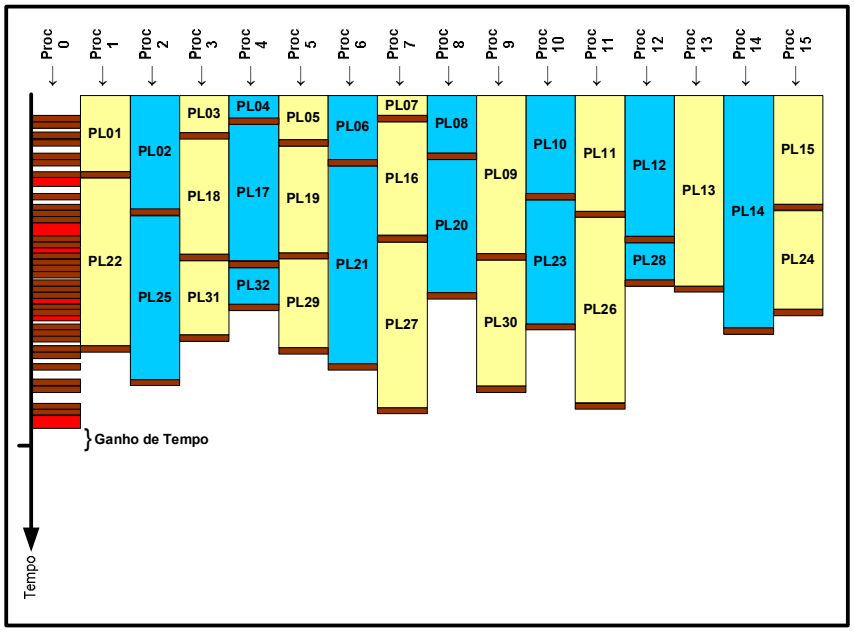

Figura 56 – Nova Forma de Agrupar os Cortes

Nesta nova estratégia de agrupamento de cortes, pode-se chegar à última mensagem com boa parte do agrupamento já executado, diminuindo o tempo de espera dos processadores e aumentando a eficiência da estratégia de paralelização.

- **Resultados da Solução Proposta**

Os tempos médios obtidos com esta versão para diversas quantidades de processadores estão apresentados na Tabela 15.

Tabela 15 - Resultados da Versão 4

Quantidade Processadores	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	88.072	-	-
2	44.359	1,99	99,27
4	22.428	3,93	98,17
8	12.343	7,14	89,19
16	6.594	13,36	83,47
32	3.714	23,71	74,10
40	3.190	27,61	69,02
64	2.268	38,83	60,67
80	2.109	41,75	52,19
128	1.610	54,71	42,75
200	1.639	53,75	26,87

Os valores dos fatores de aceleração e das eficiências estão mostrados graficamente na Figura 57. Com relação à versão 3, percebe-se um pequeno ganho de eficiência, excetuando-se o caso com 200 processadores, para os casos a partir de 32 processadores. Este ganho ficou em torno de 3% para os casos com 32 e 40 processadores e em torno de 5,5%-6% para os casos com 64 e 80 processadores. Esta nova forma de agrupamento dos cortes foi mais bem aproveitada no caso com 128 processadores, uma vez que o ganho de tempo ficou acima de 8%. Estes valores podem ser vistos na Tabela 16.

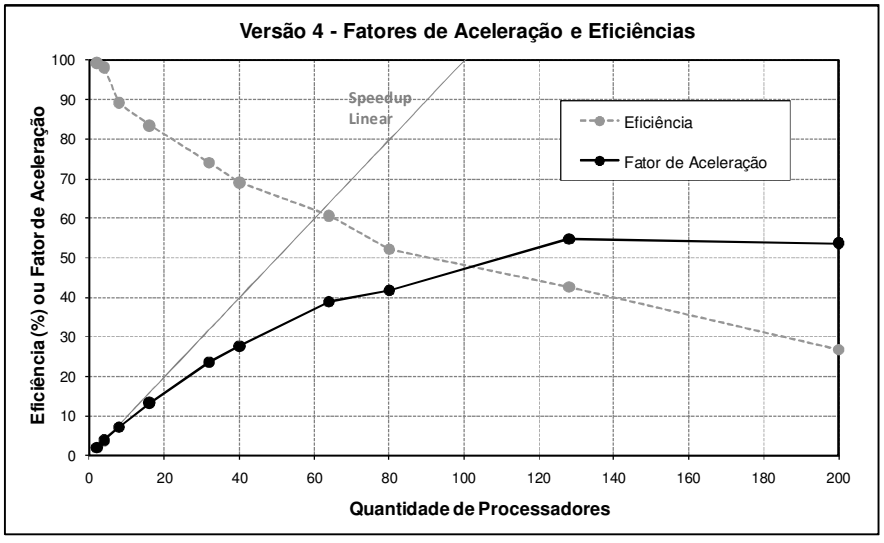

Figura 57 - Fatores de Aceleração e Eficiências da Versão 4

Tabela 16 - Comparação dos Tempos Médios entre as Versões 3 e 4

Quantidade Processadores	Versão 3 (s)	Versão 4 (s)	Diferença de Tempos (%)
1	88.173	88.072	-0,11
2	44.389	44.359	-0,07
4	22.496	22.428	-0,30
8	12.417	12.343	-0,59
16	6.648	6.594	-0,80
32	3.826	3.714	-2,91
40	3.298	3.190	-3,28
64	2.410	2.268	-5,88
80	2.236	2.109	-5,66
128	1.756	1.610	-8,33
200	1.633	1.639	0,33

Com mais estes ganhos de tempo, é importante notar que a eficiência da estratégia de paralelização, como um todo, apesar de manter a característica de saturação com o aumento da quantidade de processadores, elevou a eficiência de todos os casos em relação à versão inicial. Por exemplo, a eficiência do caso com 32 e 64 processadores, está em 74,1% e 60,7%, respectivamente, bem superior aos valores de 46,8% e de 30,7% da versão inicial.

5.4 Versão 5 - Adequação à Arquitetura de *Hardware*

O objetivo desta nova versão é aumentar a eficiência da estratégia de paralelização através da diminuição dos tempos de comunicação dos envios/recebimentos de dados. Esta diminuição se dá pela troca dos tempos mais lentos da comunicação entre as placas *blade* do *cluster*, utilizando a rede de comunicação, pelos tempos mais rápidos da comunicação entre os processadores existentes dentro das placas *blade*.

- **Identificação e Diagnóstico do Problema**

A utilização do comunicador padrão do MPI, chamado de MPI_COMM_WORLD, sempre disponível em toda aplicação MPI, possui o inconveniente de utilizar muito a rede de comunicação para realizar as operações coletivas, onde todos os processadores executam a mesma operação simultaneamente.

No exemplo mostrado na Figura 58, onde existem três placas com oito processadores cada uma, o envio de mensagens do processador 0 para os processadores

1, 4, 7, 10, 13, 16, 19 e 22, situados na segunda placa, utilizará a rede de comunicação existente entre as placas 1 e 2. O mesmo acontece com os processadores existentes na placa 3, que são o 2, 5, 8, 11, 14, 17, 20 e 23. Já a comunicação entre o processador 0 e os processadores 3, 6, 9, 12, 15, 18 e 21 não utiliza a rede de comunicação, sendo muito mais rápido o envio e recebimento de dados e resultados entre estes processadores.

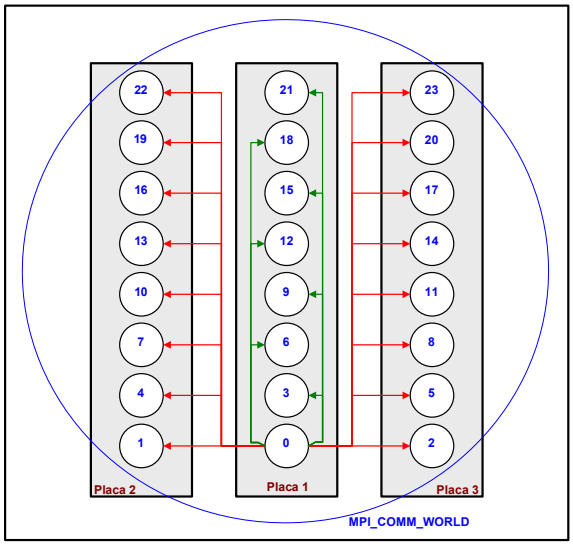

Figura 58 - *Broadcast* Utilizando o Comunicador Padrão MPI_COMM_WORLD

- **Solução Proposta**

Para diminuir a comunicação entre os processadores utilizando a rede de comunicação, objetivando uma melhoria na eficiência da estratégia de paralelização, foi implementada uma mudança nas mensagens de *broadcast*, onde um determinado processador envia dados para todos os outros. Estas mensagens, em vez de utilizar o comunicador padrão MPI_COMM_WORLD, passaram a utilizar dois comunicadores criados em tempo de execução, que são o MPI_COMM_NV1 e o MPI_COMM_NV2.

O procedimento para criação destes comunicadores é o seguinte:

(1) Determinação dos conjuntos de agrupamentos que existem no caso que acabou de ser executado pelo MPI. Estes agrupamentos são obtidos através do nome de cada processador, sendo que os que pertençam a uma mesma placa possuem os mesmos nomes. Cada processador obtém o seu respectivo nome e o envia para o processador mestre gerar os agrupamentos;

(2) De cada agrupamento é escolhido um processador, normalmente o primeiro que entrou na lista do agrupamento, para integrar um conjunto a parte. A partir deste conjunto é criado o grupo MPI_GROUP_NV1 através das funções MPI_COMM_GROUP e MPI_COMM_INCL;
(3) A partir do grupo MPI_GROUP_NV1 é criado o comunicador MPI_COMM_NV1 através da função MPI_COMM_CREATE. É importante ressaltar que somente os processadores que participarem deste grupo devem executar esta função simultaneamente;
(4) Através dos agrupamentos obtidos em (1), o comunicador MPI_COMM_NV2 é criado através da função MPI_COMM_SPLIT.

Após a criação destes comunicadores, todos os processadores de uma mesma placa estarão integrados a um comunicador MPI_COMM_NV2, ou seja, se existirem três placas com oito processadores, existirão três comunicadores MPI_COMM_NV2, um para cada placa, com oito processadores (numerados de 0 a 7). Um processador de cada agrupamento, ou placa, integrará o comunicador MPI_COMM_NV1, que conterá, para o exemplo de três placas, três processadores (numerados de 0 a 2).

Com os dois novos comunicadores criados, cada mensagem de *broadcast*, que utilizava o comunicador MPI_COMM_WORLD, é substituída por duas mensagens de *broadcast*: (1) a primeira utiliza o comunicador MPI_COMM_NV1 para disponibilizar para um dos processadores de cada placa o conjunto de dados utilizando a rede de comunicação, conforme está mostrado na Figura 59. Nesta mensagem é utilizada a função MPI_BCAST normalmente, somente substituindo o comunicador MPI_COMM_WORLD pelo comunicador MPI_COMM_NV1. É importante ressaltar que a numeração de um processador no comunicador MPI_COMM_NV1 não necessariamente é a mesma que este possui no comunicador MPI_COMM_WORLD; (2) a segunda utiliza o comunicador MPI_COMM_NV2, tendo o cuidado de determinar o número do processador que enviará a mensagem no comunicador MPI_COMM_NV2. Este processador deverá ser o mesmo que participou da mensagem anterior utilizando o MPI_COMM_NV1, conforme está mostrado na Figura 60.

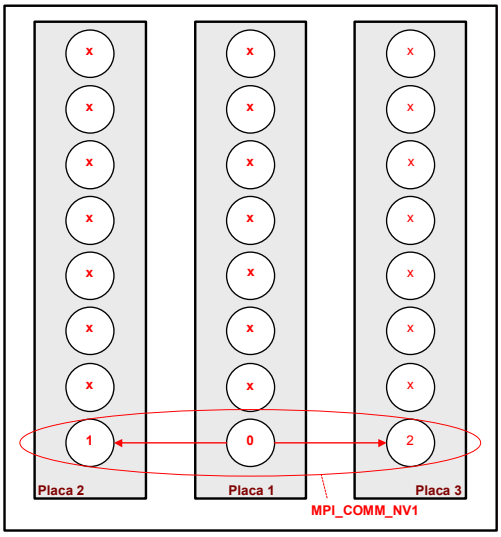

Figura 59 – Primeiro Nível de *Broadcast*

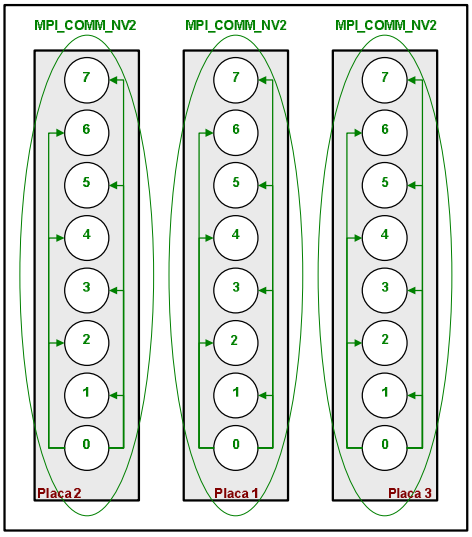

Figura 60 – Segundo Nível de *Broadcast*

Neste novo procedimento de envio de dados, a comunicação entre os processadores utilizando a rede de comunicação é significativamente diminuída. Em contrapartida, a comunicação entre os processadores de uma mesma placa, que é bem mais rápida, é significantemente aumentada.

- **Resultados da Solução Proposta**

Da mesma forma como foi feito com as versões anteriores, para avaliar esta nova versão vários casos foram executados com diversas quantidades de processadores e os seus resultados médios estão apresentados na Tabela 17.

Tabela 17 - Resultados da Versão 5

Quantidade Processadores	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	88.045	-	-
2	44.399	1,98	99,15
4	22.439	3,92	98,09
8	12.341	7,13	89,18
16	6.577	13,39	83,67
32	3.634	24,23	75,71
40	3.070	28,68	71,71
64	2.248	39,08	61,06
80	1.984	44,37	55,46
128	1.637	53,78	42,02
200	1.378	63,71	31,85

Os valores dos fatores de aceleração e das eficiências desta versão estão apresentados na Figura 61. Pode-se observar que, em relação aos fatores de aceleração,

os casos com pequenas quantidades de processadores, principalmente até 8, estão próximos à reta de *speedup* linear, indicando que as eficiências destes casos estão próximas dos 100%. Porém, a característica de saturação apresentada nos casos anteriores continua presente.

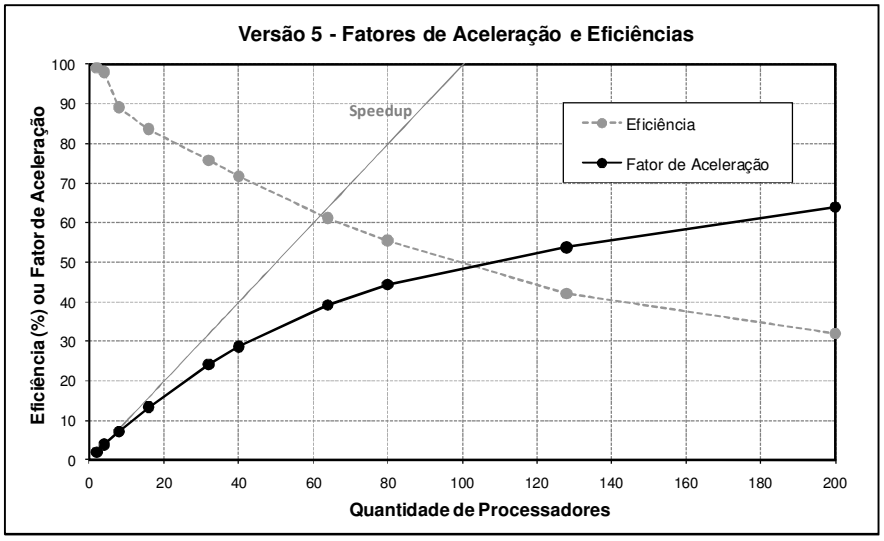

Figura 61 - Fatores de Aceleração e Eficiências da Versão 5

Com relação à versão 4, percebem-se pequenos ganhos de eficiência, excetuando-se os casos com 128 e 200 processadores: no caso com 128 processadores, ocorreu uma leve piora de 1,7%, enquanto que para o caso com 200 processadores, o ganho foi de expressivos 15,9%, conforme pode ser visto na Tabela 18.

Tabela 18 - Comparação dos Tempos Médios entre as Versões 4 e 5

Quantidade Processadores	Versão 4 (s)	Versão 5 (s)	Diferença de Tempos (%)
1	88.072	88.045	-0,03
2	44.359	44.399	0,09
4	22.428	22.439	0,05
8	12.343	12.341	-0,02
16	6.594	6.577	-0,27
32	3.714	3.634	-2,15
40	3.190	3.070	-3,77
64	2.268	2.248	-0,90
80	2.109	1.984	-5,93
128	1.610	1.637	1,70
200	1.639	1.378	-15,91

Uma grande conseqüência desta implementação é a possibilidade de futuramente se implementar uma solução híbrida com a utilização simultânea do MPI e do OpenMP,

através da manutenção do MPI nas comunicações entre as placas *blade* e a substituição do MPI pelo OpenMP nas comunicações intra placa, uma vez que o OpenMP é mais eficiente quando ocorre a utilização de memória compartilhada, que é o caso dos processadores dentro de uma placa *blade*.

5.5 Versão 6 - Compensação da Deficiência do *Hardware*

Durante a execução da versão anterior, ao se analisar a evolução das eficiências da estratégia de paralelização ao longo do processo iterativo, detectou-se uma diminuição da mesma em alguns casos. A identificação e a solução adotada para este problema estão descritas ao longo deste item.

- **Identificação e Diagnóstico do Problema**

Ao executar o caso com a versão 5, observou-se uma deterioração do desempenho ao longo das iterações quando o caso foi executado com diversas quantidades de processadores. Esta deterioração foi mais acentuada nos casos com 8 e 16 processadores, como pode ser observada na Figura 62, porém alguma queda na eficiência também foi notada nos casos com 32, 40, 64 e 80 processadores. É importante observar que nos casos com 2 e 4 processadores, as eficiências se mantiveram com pouca variação ao longo das iterações. Nos casos com 128 e 200 processadores, a variação da eficiência ao longo das iterações foi bem maior do que nos casos com 2 e 4 processadores, porém a deterioração da eficiência não foi observada, conforme está mostrado na Figura 62.

Como os casos com 2 e 4 processadores não apresentaram diminuição da eficiência e nestes dois casos não foram utilizados todos os processadores disponíveis da placa *blade* do *cluster*, desconfiou-se que o problema pudesse estar associado à utilização integral dos recursos disponíveis nesta placa. Para verificar se este problema estava ocorrendo, foram executados os casos com 8 até 128 processadores, modificando a forma de execução para reservar sempre o dobro de recursos ao que era necessário. Ou seja, na execução com 8 processadores, foram alocados 4 numa placa e 4 em outra, sendo necessário reservar todos os processadores destas 2 placas. Para o caso com 16 processadores, foram utilizados 4 processadores de 4 placas, sendo necessário reservar

todos os recursos destas 4 placas, totalizando 32 processadores. Os resultados destes casos estão apresentados na Figura 63.

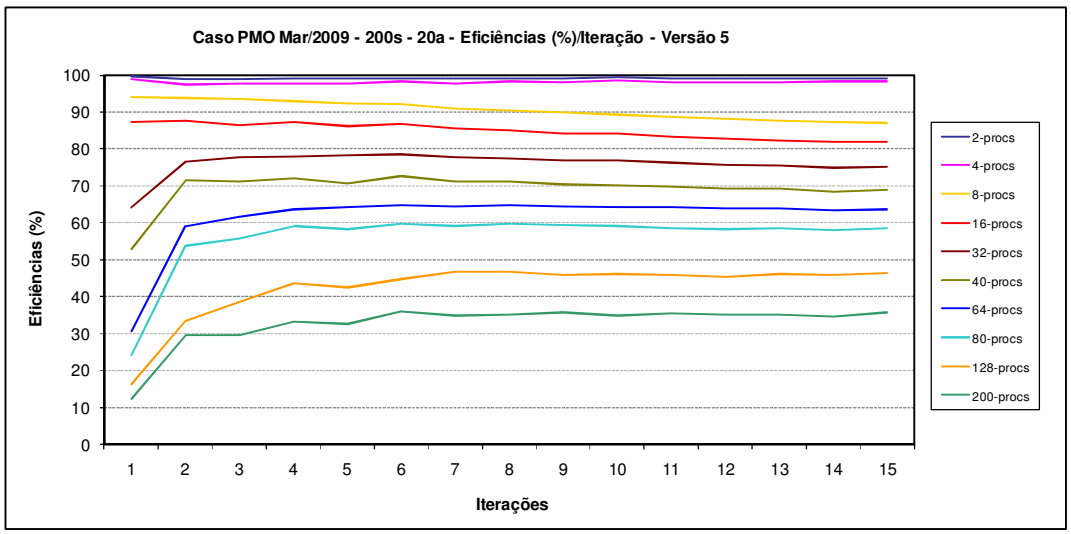

Figura 62 – Eficiências dos Casos com a Versão 5

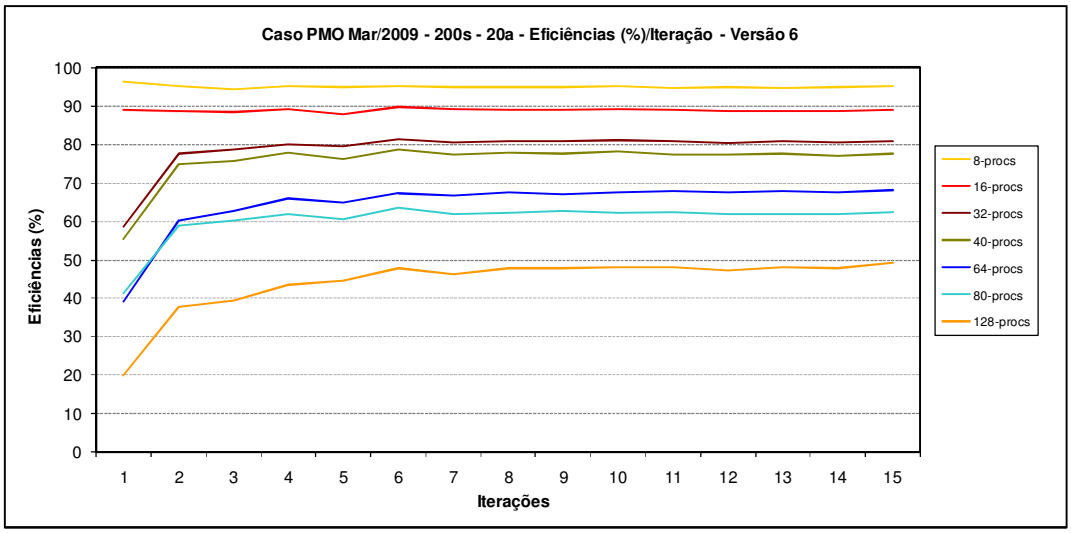

Figura 63 – Eficiências dos Casos com a Versão 6

Pode-se observar que as diminuições nas eficiências apresentadas inicialmente não foram identificadas nesta forma de executar os casos. Para comparar melhor as duas formas de execução dos casos, a Figura 64 e a Figura 65 mostram as eficiências dos casos com 8, 16, 32 e 64 processadores.

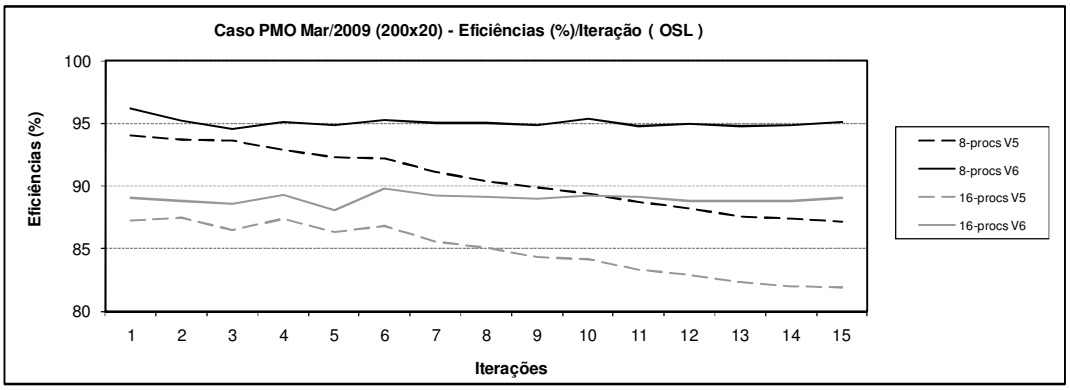

Figura 64 – Comparação entre as Versões 5 e 6 (8 e 16 Processadores)

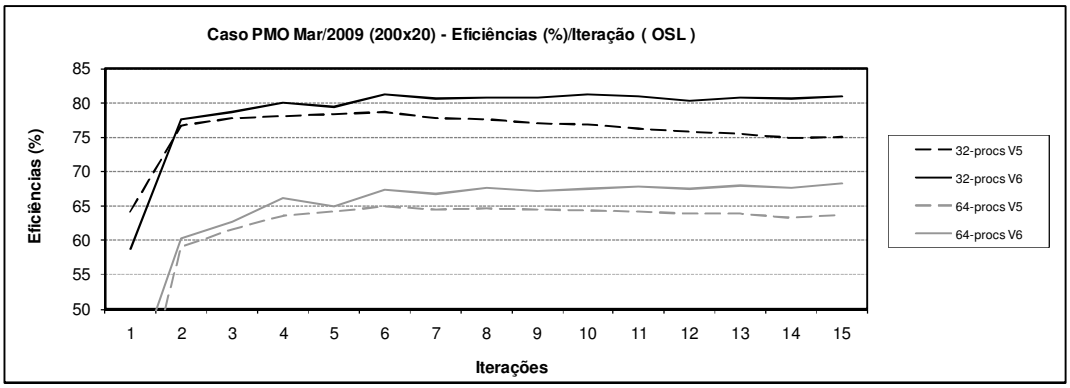

Figura 65 - Comparação entre as Versões 5 e 6 (32 e 64 Processadores)

Na Figura 64 percebe-se a grande deterioração das eficiências ao longo das iterações dos casos com 8 e 16 processadores durante a execução com a versão 5. Já no caso com a versão 6 (duplicação dos recursos disponíveis), estes casos, além de não apresentar a deterioração, ainda tiveram as eficiências melhoradas nas iterações iniciais. Com relação à Figura 65, para o caso com 32 processadores observa-se o mesmo efeito ocorrido nos casos com 8 e 16 processadores, que foi a diminuição da eficiência ao longo do processo iterativo. Para o caso com 64 processadores, a queda na eficiência foi muito pouco notada na simulação com a versão 5. Já no caso com a versão 6, a eficiência apresentou uma tendência de alta ao longo das iterações.

A próxima etapa é identificar quais são as razões que estão fazendo com que as eficiências estejam diminuindo na execução normal e estejam se mantendo na execução com o dobro de recursos de processamento. A primeira possibilidade a ser cogitada foi um possível problema de acesso a disco. Como na versão 2 foi feita a troca de comunicação por armazenamento local dos cortes e como cada processo realiza acessos

de leitura/escrita neste arquivo quase simultaneamente ao longo da execução do caso, estes 8 intensos acessos poderiam estar gerando uma contenção na leitura ou na escrita destes arquivos. Para averiguar se esta desconfiança é real, foram inseridas instruções de tomada de tempo para identificar os tempos gastos ao longo da rotina *backward*. Estes tempos foram obtidos para o caso de PMO de Março de 2009 com um aumento para 50 na quantidade de cenários de afluência das aberturas do ciclo *backward* para aumentar o tempo de execução e evidenciar ainda mais a diferença dos tempos entre as duas formas de execução. Os tempos obtidos estão disponíveis para consulta no item B.3 do Anexo B.

Dos resultados destes testes, mostrados na Tabela 52 e na Tabela 53, podem-se tirar algumas conclusões interessantes: os tempos correspondentes às soluções de diversas séries hidráulicas e suas respectivas aberturas indicam uma boa distribuição de carga entre os processadores. A Tabela 52 mostra estes tempos variando de 38,29s até 41,25s, e na Tabela 53 estes tempos variam de 33,04s até 35,98s; uma boa parcela do tempo consumido na tarefa de envio/recebimento dos cortes corresponde ao tempo ocioso dos processadores, uma vez que este é o único ponto de sincronismo, pois todos os processadores precisam esperar a liberação da comunicação.

A diferença entre os tempos das duas formas de execução do caso aconteceu nas tarefas referente à montagem, solução e obtenção dos resultados dos PLs, que é onde se concentra a maior parte das tarefas a serem executadas na rotina *backward*. Este fato elimina a suspeita inicial de que a diferença seria de algum problema de leitura/escrita no arquivo de cortes, pois os tempos consumidos nestas etapas não tiveram diferenças significativas. Logo, a diferença dos tempos destas duas execuções, 5,29s, não pode ser explicada pelos tempos gastos nestas tarefas.

O grande responsável pela diferença foram as tarefas referentes à montagem, solução e obtenção dos resultados dos PLs, só que estas são as mesmas nas duas maneiras de executar o caso, não havendo, a princípio, razão que justificasse a diferença de tempo entre os dois casos. As suspeitas passam a cair sobre o processador, propriamente dito, uma vez que ele é o responsável pela execução destas tarefas.

Elias, Camata *et al.* [143] apresentaram uma análise sobre o desempenho de processadores de famílias Intel Xeon anteriores à atual família, chamada de *Nehalem*. Os casos deste trabalho foram executados nos seguintes processadores: Intel Xeon

Quad-Core, *Clovertown*, X5355; Intel Xeon *Quad-Core*, *Harpertown*, E5450; e Intel Xeon *Quad-Core*, *Nehalem*, X5560. Os autores identificaram que ao executar o programa EdgeCFD em sistemas compostos por grupos de dois processadores *Quad-Core*, os dois processadores mais antigos apresentaram uma eficiência significativamente pior do que na execução no sistema com o processador *Nehalem*. Para compensar a queda na eficiência do programa, foi necessário utilizar uma quantidade maior de processadores nos sistemas com as famílias anteriores do processador Intel Xeon. A conclusão dos autores indica claramente que estes processadores *Quad-Core* de famílias anteriores perdem dramaticamente a sua eficiência quando utilizados de forma intensa em todos os seus núcleos de processamento.

Como os processadores que estão sendo utilizados nas placas *blades* do *cluster* são do tipo *Clovertown*, a conclusão que se chega é que a deterioração da eficiência da estratégia é causada pelo processador que está sendo utilizado e não por uma deficiência da mesma propriamente dita.

- **Solução Proposta**

Para este problema não é possível arrumar uma solução que implique em alteração da estratégia de paralelização, pois se trata de uma característica da família dos processadores utilizados (*hardware*). A recomendação a ser dada é reservar o dobro de recursos computacionais para minimizar a deterioração da eficiência da estratégia de paralelização.

- **Resultados da Solução Proposta**

Executando os mesmos casos, com a mesma estratégia definida na versão 5, porém utilizando o dobro de recursos computacionais, obtiveram-se os valores mostrados na Tabela 19.

Tabela 19 - Resultados da Versão 6

Quantidade Processadores	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	88.045	-	-
8	11.591	7,60	94,95
16	6.193	14,22	88,86
32	3.462	25,43	79,47
40	2.905	30,31	75,77
64	2.124	41,45	64,76
80	1.841	47,83	59,79
128	1.555	56,63	44,24

Com a reserva do dobro de processadores, os casos com 2, 4 e 200 processadores não foram executados. Os casos com 2 e 4 porque não apresentaram deterioração do desempenho ao longo da execução com a versão 5. O caso com 200 processadores não pôde ser executado porque o *cluster* não possui recursos computacionais suficientes para reservar o dobro da capacidade necessária para este caso.

Os resultados dos fatores de aceleração e das eficiências estão mostrados na Figura 66.

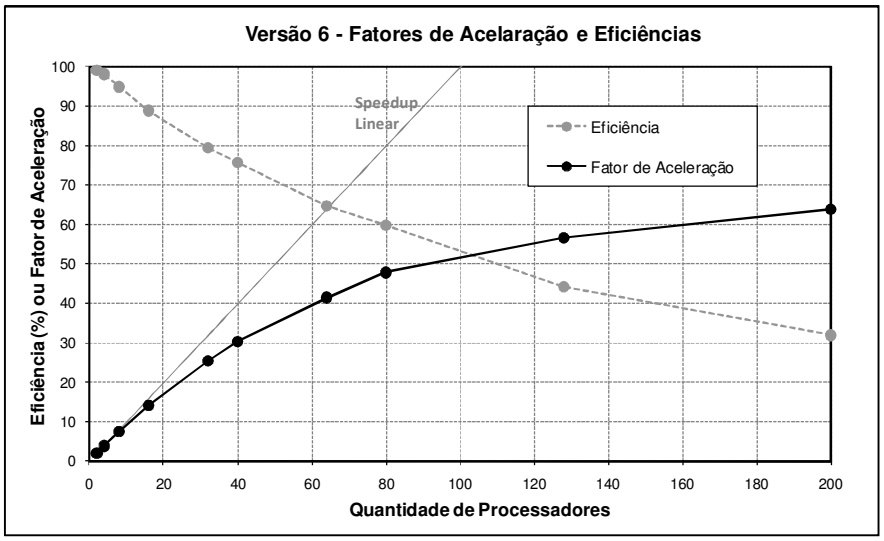

Figura 66 - Fatores de Aceleração e Eficiências da Versão 6

De posse dos tempos desta nova maneira de executar os casos, pode-se compará-los com os obtidos na maneira tradicional, como mostra a Tabela 20.

Tabela 20 - Comparação dos Tempos Médios entre as Versões 5 e 6

Quantidade Processadores	Versão 5 (s)	Versão 6 (s)	Ganho de Tempo (%)
8	12.341	11.591	6,07
16	6.577	6.193	5,83
32	3.634	3.462	4,74
40	3.070	2.905	5,36
64	2.248	2.124	5,50
80	1.984	1.841	7,24
128	1.637	1.555	5,03

Na comparação entre esta versão e a anterior, pode-se observar que todos os casos apresentaram uma redução de tempo final, que foi de 4,74%, para o caso com 32 processadores, até 7,24% para o caso com 80 processadores. Como a ordem de grandeza das reduções foram as mesmas, não se pode afirmar que o procedimento de reservar o dobro da quantidade de processadores tenha sido mais favorável a um determinado caso. Pelo contrário, a conclusão que pode ser tirada dos resultados é que o problema seria realmente sistêmico, beneficiando todos os casos de forma muito parecida.

O ideal é poder testar a versão final da estratégia de paralelização numa nova família de processadores, que não possua as restrições apresentadas pelos existentes no *cluster* em que os casos foram executados, e verificar se as eficiências obtidas com a reserva do dobro da quantidade de processadores possam ser reproduzidas sem a necessidade desta reserva extra.

5.6 Análise do Resultado da Otimização

Uma ótima forma de comparar as eficiências de todas as seis versões que foram geradas é colocá-las num mesmo gráfico, conforme está apresentado na Figura 67. Neste gráfico todos os valores de eficiências foram colocados na mesma ordem em que as respectivas versões foram geradas, fornecendo uma idéia muito boa da evolução que foi obtida com as análises dos itens anteriores deste capítulo.

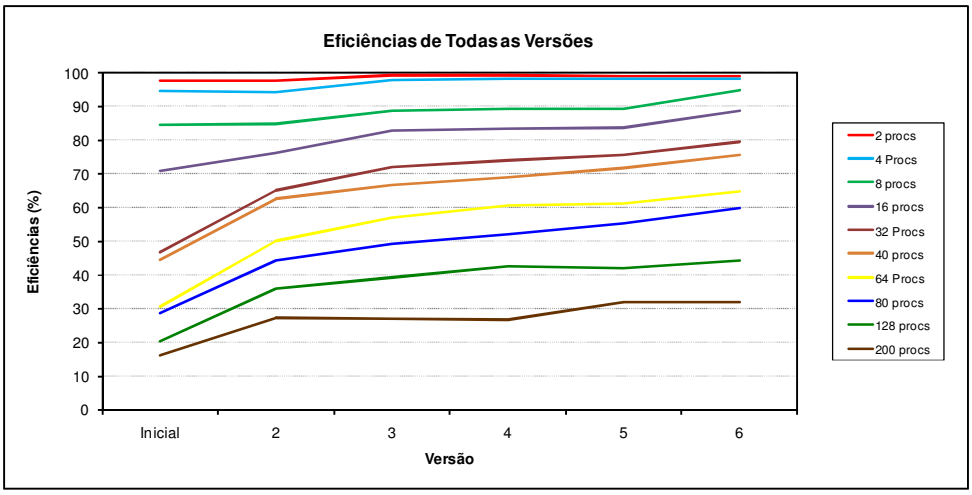

Figura 67 – Eficiências (%) de Todas as Versões

As eficiências dos casos com 2 e 4 processadores se mantiveram sempre com valores bastante elevados, muito próximos da eficiência máxima de 100%. Já os casos com 8 e 16 processadores apresentaram eficiências iniciais não tão elevadas, por volta de 85% para o caso de 8 processadores e pouco mais de 70% para o caso com 16 processadores. Com as diversas análises e implementações, estas eficiências alcançaram valores bastantes elevados, acima de 90% para o caso com 8 e quase 90% para o caso com 16 processadores. Os casos com 32 e 40 processadores apresentaram evolução bem similar, com eficiências um pouco melhores para o caso com 32. Nestes casos, as eficiências iniciais com valores por volta de 45% foram aumentadas para valores entre 75% e 80%. Os casos com 64 e 80 processadores também apresentaram grandes aumentos das suas eficiências, saindo de valores por volta de 30% para valores em torno dos 60%. O aumento das eficiências do caso com 128 processadores também foi bastante significativa, apesar do valor final ter ficado por volta de 45%, um valor que não é elevado. Mas é importante ressaltar que um caso de PMO possui 200 séries hidrológicas e a execução com 128 processadores acaba não permitindo uma boa distribuição de carga entre os processadores, provocando uma eficiência abaixo de 50%. O último caso corresponde ao de 200 processadores, no qual a distribuição dos problemas pelos processadores é dependente apenas da série hidrológica que o processador recebeu. Evidentemente que a eficiência neste caso não poderia ser alta, mas mesmo com as condições restritivas, em termos de possibilidade de melhoria de eficiência, o valor final foi o dobro do inicial, passando dos 16% para 32%.

A visualização gráfica das eficiências também pode ser vista na Figura 68, onde se pode reparar nos ganhos entre as seis versões da estratégia de paralelização.

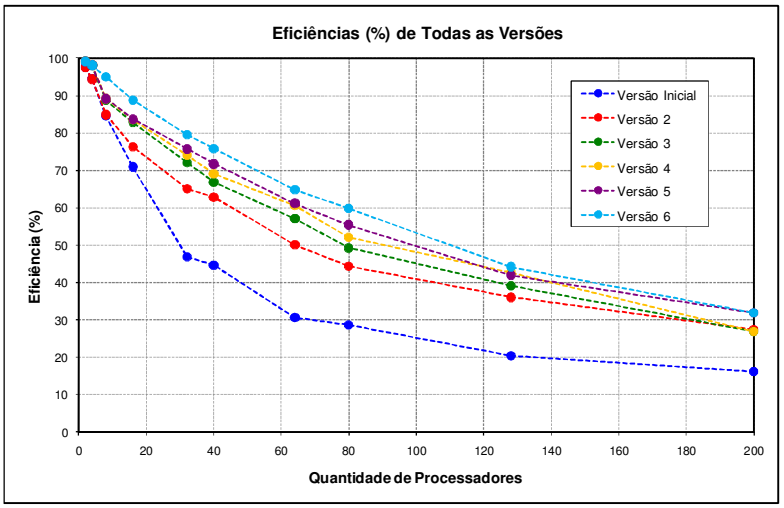

Figura 68 – Eficiências das Versões de Estratégia de Paralelização

Com relação aos fatores de aceleração, a aproximação dos valores à reta de *speedup* linear é claramente visível na Figura 69, principalmente para os casos com quantidades menores de processadores.

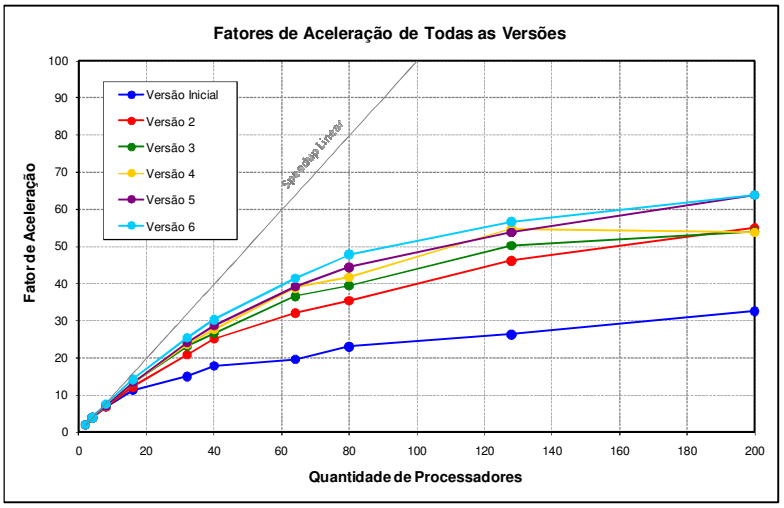

Figura 69 – Fatores de Aceleração das Versões de Estratégia de Paralelização

Os resultados apresentados mostram que o trabalho desenvolvido permitiu que a eficiência da estratégia de paralelização fosse sensivelmente melhorada, com um conseqüente aumento nos fatores de aceleração em relação à primeira estratégia de paralelização desenvolvida.

Capítulo 6.
Resultados Finais

Neste capítulo estão apresentados os resultados obtidos pela última versão da metodologia proposta para aplicação de processamento paralelo no problema de planejamento da operação de sistemas hidrotérmicos. Tal qual ocorrido no capítulo anterior, neste também serão apresentados os tempos médios das execuções. Caso seja desejado consultar algum dos tempos dos casos executados, os mesmos estão disponibilizados no Anexo C

6.1 Desempenho da Estratégia de Paralelização Utilizando a Biblioteca COIN

Durante toda a etapa de melhoria da eficiência da estratégia de paralelização, a biblioteca OSL foi utilizada para resolver os PLs da aplicação. Esta biblioteca é um pacote comercial, logo, o acesso ao seu código fonte não é possível, e, por ter sido descontinuada pela IBM, só possui a versão para sistemas operacionais de 32 bits, não permitindo a compilação da aplicação e, conseqüentemente, não permitindo a avaliação da estratégia de paralelização em um sistema operacional de 64 bits.

A biblioteca CLP (COIN *Linear Programming*) é um dos projetos do COIN-OR (*Computational Infrastructure for Operations Research*) e consiste num pacote de solução de problemas de programação linear utilizando os algoritmos primal e dual simplex. O código fonte deste pacote é disponibilizado gratuitamente (*open source*) e, depois de compilado, se torna uma biblioteca que pode ser utilizada pela aplicação onde foi implementada a estratégia de paralelização.

A utilização da biblioteca COIN permite que a estratégia de paralelização possa ser analisada com outro pacote de solução de programação linear e, o mais importante, também permite a realização de testes em ambiente de 64 bits.

Nestas novas versões, por causa da utilização da biblioteca COIN, suas execuções apresentarão resultados diferentes dos casos executados com a biblioteca OSL, sendo necessária uma etapa de validação dos resultados, conforme será apresentado a seguir.

6.1.1 Análise das Convergências

Para validar as duas versões com a biblioteca COIN, o caso de PMO de março de 2009 foi utilizado como base, e os resultados destas novas versões foram comparados com os obtidos pela versão com a biblioteca OSL. Os valores da última iteração de cada um dos três processos de convergência estão apresentados na Tabela 21. As três convergências completas estão disponibilizadas na Tabela 54, situada no item B.4 do Anexo B. Os três processos iterativos convergiram para limites máximos e mínimos do custo de operação muito próximos, além disto, as trajetórias das três convergências também foram bastante parecidas, sendo que o processo levou 13 iterações na versão com a biblioteca COIN 64 bits e 15 para as outras duas. A maior diferença entre os valores de Zinf ocorreu entre as versões COIN 32 bits e COIN 64 bits e foi de 0,52% (51.649,4 e 51.915,6) e a maior diferença entre os valores de Zsup, por volta de 0,14%, ficou entre a versão com OSL e a versão com a COIN 32 bits (62.421,8 e 62.333,2). Pela análise destas convergências, pode-se afirmar que, para este caso, as três versões apresentaram resultados estatisticamente semelhantes.

Tabela 21 – Valores Finais das Convergências do Caso PMO Março de 2009 (Valores em 10^6 R$)

Caso PMO 2010/04					
Biblioteca	Iteração Final	Limite Inferior	Zinf	Limite Superior	Zsup
OSL	15	51.615,4	51.772,1	71.804,0	62.421,8
COIN 32 bits	15	51.736,4	51.915,6	71.740,6	62.333,2
COIN 64 bits	13	51.505,4	51.649,4	71.855,1	62.389,3

As três trajetórias de convergência podem ser vistas nas seguintes figuras: na Figura 70 para o caso com a biblioteca OSL; na Figura 71 para o caso com a biblioteca COIN 32 bits; e na Figura 72 estão apresentados os valores com a biblioteca COIN 64 bits. Nas três figuras estão apresentadas em detalhe as partes finais das respectivas convergências, podendo-se confirmar as similaridades nas trajetórias das três soluções.

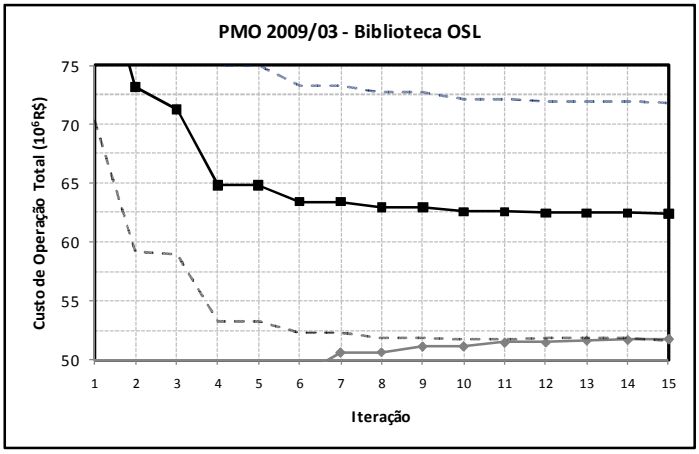

Figura 70 - Processo Iterativo do Caso PMO Março de 2009 com Biblioteca OSL

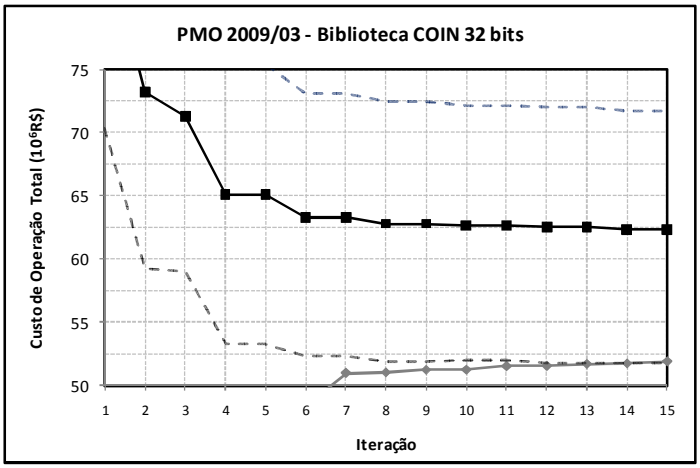

Figura 71 - Processo Iterativo do Caso PMO Março de 2009 com Biblioteca COIN 32 bits

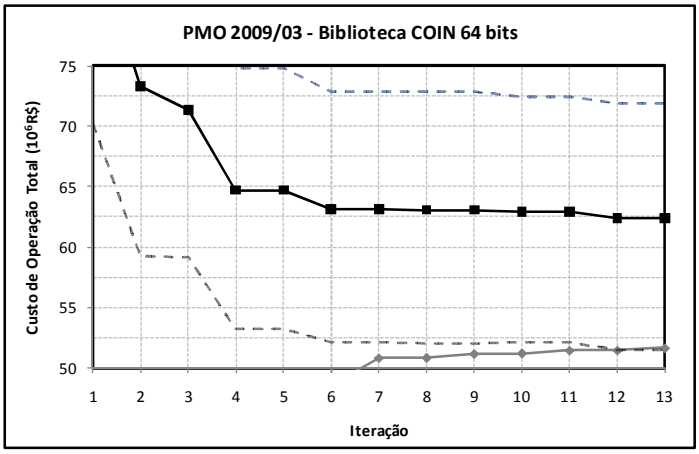

Figura 72 – Processo Iterativo do Caso PMO Março de 2009 com Biblioteca COIN 64 bits

6.1.2 Análise das Eficiências

A partir do instante em que os resultados das versões com diferentes bibliotecas de solução de PLs estão coerentes entre si, pode-se passar à etapa de análise das eficiências. Para obter estes resultados, cada uma das três bibliotecas foi utilizada com as duas últimas versões obtidas no capítulo anterior, totalizando seis conjuntos de resultados, conforme está apresentado na Tabela 22. Convém ressaltar que os valores referentes à biblioteca OSL são os mesmos já apresentados na Tabela 17 e na Tabela 19 e estão repetidos aqui para facilitar a comparação dos resultados com os das bibliotecas COIN.

Este caso demanda um tempo muito longo para convergir seqüencialmente (utilização de 1 processador): utilizando a biblioteca OSL, o processo de convergência precisou de 15 iterações e 24h 27min para solucionar o caso; a biblioteca COIN 32 bits foi mais lenta ainda, pois, com as mesmas 15 iterações, precisou de 27h 6min para terminar o processamento; no caso da biblioteca COIN 64 bits, percebe-se que o caso foi resolvido bem mais rapidamente, já que a execução levou 14h 51min para terminar. Apesar do processo de convergência ter levado duas iterações a menos, a diferença de 9h 36min é superior ao tempo que provavelmente seria consumido nestas iterações adicionais, indicando que a versão com a biblioteca COIN 64 bits foi realmente a mais rápida das três.

Tabela 22 – Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 para as Três Opções de Bibliotecas de Solução de PLs

Versão	Qte. Procs.	Biblioteca OSL			Biblioteca COIN 32 bits			Biblioteca COIN 64 bits		
		Tempo Médio (s)	Fator de Aceleração	Eficiência (%)	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
5	1	88.045	-	-	97.560	-	-	53.509	-	-
	2	44.399	1,98	99,15	49.952	1,95	97,65	27.543	1,94	97,14
	4	22.439	3,92	98,09	25.482	3,83	95,72	14.076	3,80	95,04
	8	12.341	7,13	89,18	13.946	7,00	87,45	8.020	6,67	83,40
	16	6.577	13,39	83,67	7.396	13,19	82,44	4.240	12,62	78,87
	32	3.634	24,23	75,71	4.075	23,94	74,82	2.362	22,66	70,80
	40	3.070	28,68	71,71	3.427	28,47	71,17	2.004	26,70	66,75
	64	2.248	39,17	61,20	2.502	39,00	60,93	1.496	35,77	55,89
	80	1.984	44,37	55,46	2.151	45,36	56,69	1.288	41,54	51,93
	128	1.637	53,78	42,02	1.745	55,91	43,68	1.091	48,93	38,23
	200	1.378	63,89	31,95	1.570	62,16	31,08	907	58,97	29,49
6	8	11.591	7,60	94,95	13.108	7,44	93,04	7.246	7,38	92,31
	16	6.193	14,22	88,86	6.969	14,00	87,50	3.845	13,92	86,97
	32	3.462	25,43	79,47	3.878	25,16	78,62	2.171	24,64	77,01
	40	2.905	30,31	75,77	3.260	29,93	74,82	1.829	29,26	73,14
	64	2.124	41,45	64,76	2.362	41,31	64,55	1.367	39,13	61,15
	80	1.841	47,83	59,79	2.088	46,73	58,41	1.164	45,98	57,48
	128	1.555	56,63	44,24	1.653	59,02	46,11	1.017	52,61	41,10

Os valores dos fatores de aceleração e das eficiências estão apresentados graficamente na Figura 73 e na Figura 74, respectivamente.

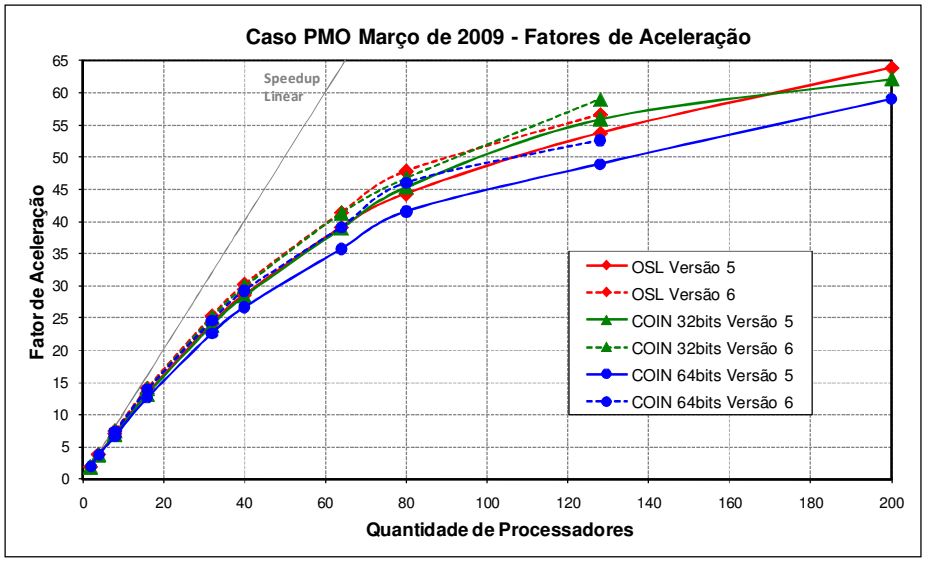

Figura 73 – Fatores de Aceleração do Caso PMO de Março de 2009 com as Versões 5 e 6 e as Três Opções de Bibliotecas de Solução de PLs

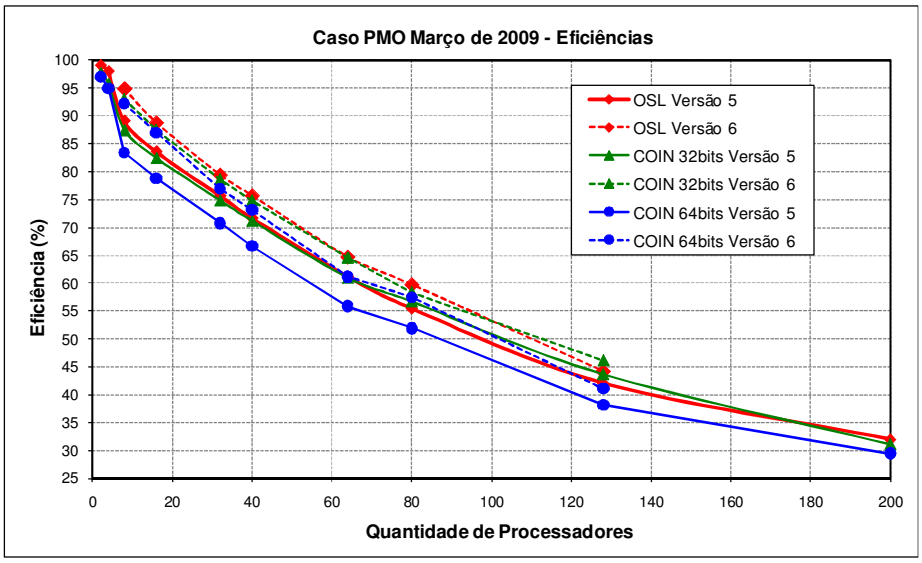

Figura 74 – Eficiências do Caso PMO de Março de 2009 com as Versões 5 e 6 e as Três Opções de Bibliotecas de Solução de PLs

Como a análise dos resultados da versão com a biblioteca OSL já foi feita nos itens 5.4 e 5.5, agora serão feitas apenas as análises dos casos com as duas versões das bibliotecas COIN, além de uma comparação destas versões com os obtidos com a biblioteca OSL.

A análise dos tempos de execução mostram que a biblioteca COIN 64 bits foi a mais rápida em qualquer das quantidades de processadores, seja na versão 5 ou na 6. Por exemplo, no caso com 128 processadores, a versão 6 levou apenas 1007s, ou seja, 16min 57s para resolver o caso, com uma eficiência de 41,10%. Sob a ótica puramente da eficiência, este valor pode ser dito baixo, porém, saber que um caso real, que demanda quase 15 horas seqüencialmente, poder ser executado em apenas 17 minutos é um fato bastante relevante.

Analisando puramente os números das eficiências e dos fatores de aceleração, pode-se observar que os casos com a biblioteca COIN 32 bits obtiveram valores melhores do que os da biblioteca COIN 64 bits e ambos foram, em geral, piores do que os da biblioteca OSL. Compondo estes resultados com os tempos decorridos, verifica-se inicialmente que a biblioteca COIN 32 bits levou mais tempo e teve, na maioria dos casos, um desempenho pouco pior que a da biblioteca OSL. Este fato só não ocorreu nos casos da versão 5 com 80 e 128 processadores e no caso da versão 6 com 128 processadores, onde a eficiência da biblioteca COIN 32 bits foi um pouco melhor do que a da biblioteca OSL.

Com relação aos valores um pouco piores das eficiências obtidas com a biblioteca COIN 64 bits, não se pode esquecer que as suas execuções foram as mais rápidas. Deve-se ressaltar que a solução do problema de planejamento da operação utilizando a PDDE possui um ponto de sincronismo em cada período de cada iteração, por conta do agrupamento dos cortes de Benders, e esta etapa é um conjunto de operações de execução serial na estratégia de paralelização. Logo, o tempo consumido neste agrupamento influi diretamente no desempenho da estratégia, uma vez que a sua execução, ao diminuir o tempo de processamento dos casos através da divisão das séries em vários processadores e com o aumento da capacidade de processamento do processador, fica relativamente mais significativa. Como na etapa de agrupamento existe muito acesso a disco, através de leituras e escritas em arquivo, o desempenho de toda a estratégia de paralelização fica dependente do tempo de acesso à disco. Presume-se que este tempo não tenha variado significativamente em nenhum dos casos, uma vez que os discos foram os mesmos e não sofreram significativas mudanças do espaço total ocupado durante as simulações. Logo, como os tempos de processamento dos casos com a biblioteca COIN 64 bits foram menores, é razoável esperar que as eficiências sejam

um pouco menores e pelos resultados das versões 6, este desempenho um pouco pior é amplamente compensado pelo significativo ganho de tempo final de processamento.

Com relação aos tempos das duas bibliotecas de 32 bits, como os tempos com a biblioteca OSL foram quase sempre menores e as eficiências foram quase sempre melhores, conclui-se que esta biblioteca é, na maioria das quantidades de processadores deste caso, mais eficiente que a COIN.

Partindo para a comparação dos resultados entre as versões 5 e 6 das duas bibliotecas COIN, observa-se a mesma característica de piora da eficiência com o aumento da quantidade de processadores que ocorreu com a biblioteca OSL. As diferenças destas eficiências com a biblioteca COIN 32 bits foram bastante parecidas com as da biblioteca OSL, como podem ser vistas na Tabela 23, exceção apenas para o caso com 80 processadores, onde o ganho de tempo foi de 7,24% com a biblioteca OSL e de apenas 2,94% com a biblioteca COIN 32 bits. Já o mesmo não pode ser dito com relação à biblioteca COIN 64 bits, onde as diferenças dos tempos foram, em geral, um pouco maiores que nas duas versões com 32 bits. Enquanto que os ganhos de tempo ficaram entre 2,94% e 6,01% para a biblioteca COIN 32 bits, os ganhos ficaram entre 6,81% e 9,66% com a utilização da biblioteca COIN 64 bits. Assumindo que, com a utilização da versão 6, os problemas dos processadores da família Intel *Core2Quad* são minimizados, com o ônus de se reservar o dobro da capacidade computacional, um ganho maior de tempo significa que a versão 5 foi mais prejudicada. Logo, os ganhos de tempo maiores obtidos com a biblioteca COIN 64 bits, indicam o problema existente nos processadores da família Intel *Core2Quad* foi mais prejudicial às execuções da versão 5 desta biblioteca.

Tabela 23 – Ganhos de Tempo do Caso PMO de Março de 2009 para as Três Opções de Bibliotecas de Solução de PLs Utilizando as Versões 5 e 6

Ganhos de Tempo entre as Versões 5 e 6 (%)			
Qte. Procs.	OSL	COIN 32bits	COIN 64bits
8	6,07	6,01	9,66
16	5,83	5,78	9,32
32	4,74	4,83	8,06
40	5,36	4,88	8,73
64	5,50	5,60	8,60
80	7,24	2,94	9,65
128	5,03	5,27	6,81

Para fazer uma análise mais detalhada dos tempos de execução das três versões com bibliotecas diferentes, não é possível olhar unicamente para os números finais dos casos, uma vez que a quantidade de iterações do caso com a biblioteca COIN 64 bits foi diferente do que as das outras duas bibliotecas. Logo, para realizar esta análise, aproveitaram-se os tempos da última iteração comum dos três casos, conforme pode ser visto na Tabela 55, apresentada no item B.4 do Anexo B.

De acordo com os resultados, a execução da versão 6 com a biblioteca COIN 32 bits foi em média 13,8% mais lenta que a versão com a biblioteca OSL, tanto na versão 5 quanto na versão 6, ou seja, as diferenças entre as versões se mantiveram proporcionais nas duas versões. É interessante notar que os tempos de execução com a versão 6 da biblioteca COIN 32 bits foram piores que os da versão 5 com a biblioteca OSL. No caso da versão com a biblioteca COIN 64 bits, a redução média do tempo de processamento em relação à versão OSL foi de 15,8%, com as versões 5, e 17,2% com a utilização das versões 6 da estratégia de paralelização.

É importante ressaltar a diferença dos tempos de execução entre as duas versões, 32 e 64 bits, das bibliotecas COIN. Como o código fonte destas bibliotecas é o mesmo e o compilador GNU também é o mesmo (versão 4.1.2 20080704), com as mesmas opções de compilação, a diferença está apenas no tamanho da palavra. Isso significa que o ganho de tempo foi causado unicamente por conta do código ser de 64 bits, mostrando que o simples fato de recompilar um código em 64 bits pode aumentar o desempenho em pelo menos 25%.

Diante dos tempos de execução obtidos com a biblioteca COIN 64 bits serem bem menores do que a COIN 32 bits, o que, aliás, era o objetivo ao testar-se a biblioteca COIN em comparação à OSL, os próximos resultados conterão apenas os valores para as bibliotecas OSL e COIN 64 bits.

O objetivo dos próximos resultados é avaliar a estratégia de paralelização com a modificação de alguns parâmetros do caso base e também com outros casos de PMO. Além disto, é importante manter as simulações com as versões 5 e 6 para ter a noção do impacto da perda de desempenho com os processadores da família *Core2Quad* nestas simulações. Para tanto, serão comparados os desempenhos obtidos com 8, 16, 32, 64 e 128 processadores, comum a ambas as versões, e 200 processadores só para a versão 5.

6.2 Desempenho da Estratégia de Paralelização com a Ampliação do Caso

Quando se planeja uma estratégia de paralelização de um algoritmo, uma conseqüência da redução significativa do tempo de processamento é a possibilidade de se executar casos com alguns parâmetros aumentados. Existe uma expectativa que a estratégia ora definida melhore a eficiência quando for executada com casos de solução mais longa, em termos de processamento. Esta expectativa é baseada nos seguintes fatos:

- Além das etapas de montagem e solução dos PLs, as outras tarefas que possuem algum tempo de processamento significante são a leitura, o agrupamento e o armazenamento dos cortes. É esperado que estas tarefas não aumentem seus respectivos tempos de processamento na mesma proporção do aumento dos tempos das tarefas que dizem respeito à montagem e solução dos PLs. Como estas tarefas são feitas de forma serial, prejudicando o desempenho da estratégia de paralelização, a diminuição relativa deste tempo de processamento faz com que aumente a eficiência da estratégia adotada;
- O aumento da quantidade de séries hidrológicas faz com que a distribuição de carga dinâmica adotada tende a ser mais eficiente, em relação a um mesmo caso executado com menos séries e com a mesma quantidade de processadores.

Por conta desta expectativa, neste item serão analisadas as eficiências do caso PMO de Março de 2009, combinando o aumento da quantidade de séries hidrológicas, de 200 para 300, e o aumento da quantidade de aberturas de cada série hidrológica, de 20 para 50, resultando em três casos diferentes do original. Para cada um destes casos, serão analisados os resultados obtidos com a utilização da biblioteca OSL e com a versão de 64 bits da biblioteca COIN.

6.2.1 Caso PMO de Março de 2009 com 300 Séries

Para avaliar a estratégia de paralelização perante um aumento de parâmetros do caso PMO de março de 2009, a primeira opção foi aumentar a quantidade de séries hidrológicas de 200 para 300, mantendo a quantidade de aberturas utilizadas no ciclo

backward em 20. Os valores obtidos para as duas bibliotecas de solução de PLs, para as versões 5 e 6, estão mostrados na Tabela 24.

Tabela 24 – Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 (300 Séries x 20 Aberturas) para as Bibliotecas de Solução de PLs OSL e COIN 64 bits

Versão	Qte. Procs.	Biblioteca OSL			Biblioteca COIN		
		Tempo (s)	Fator de Aceleração	Eficiência (%)	Tempo (s)	Fator de Aceleração	Eficiência (%)
5	1	218.717	-	-	152.186	-	-
	8	31.318	6,98	87,30	23.524	6,47	80,87
	16	16.362	13,37	83,55	12.254	12,42	77,62
	32	8.728	25,06	78,31	6.571	23,16	72,38
	64	5.025	43,53	68,01	3.769	40,38	63,10
	128	3.270	66,89	52,25	2.463	61,79	48,27
	200	2.690	81,32	40,66	1.973	77,12	38,56
6	8	28.474	7,68	96,02	20.380	7,47	93,34
	16	14.945	14,63	91,47	10.690	14,24	88,98
	32	8.023	27,26	85,20	5.807	26,21	81,89
	64	4.629	47,25	73,83	3.338	45,59	71,23
	128	3.019	72,44	56,59	2.204	69,05	53,95

Os primeiros resultados a destacar foram os tempos de execução, onde a simulação com a biblioteca COIN levou 42h 16min para convergir a solução em 15 iterações, contra 60h 45min da simulação com a biblioteca OSL com 16 iterações. Apesar das eficiências maiores com a biblioteca OSL, os tempos de execução foram significativamente menores com a biblioteca COIN.

A comparação das eficiências deste caso com os obtidos no PMO de Março de 2009 original, adotado como base neste trabalho, que possui 200 séries hidrológicas, gerou os valores apresentados na Tabela 25. Nesta tabela, valores negativos indicam eficiências maiores no caso PMO original e valores positivos indicam eficiências maiores para o caso com 300 séries. Nas simulações com a biblioteca OSL, os resultados mostram que somente os casos com 8 e 16 processadores apresentaram pequenas reduções de eficiência, de 2,11% e 0,15%, respectivamente. Com todas as outras quantidades de processadores, o caso com 300 series hidrológicas apresentou eficiências superiores, apresentando ganhos de 3,44%, para 32 processadores, até 27,64%, para 200 processadores.

Tabela 25 – Diferenças (%) das Eficiências entre os Casos com 300 Séries e 200 Séries

Versão	Qte. Procs	OSL	COIN
5	8	-2,11	-3,03
	16	-0,15	-1,59
	32	3,44	2,22
	64	11,38	12,90
	128	24,36	26,02
	200	27,64	30,77
6	8	1,12	1,12
	16	2,94	2,31
	32	7,20	6,34
	64	14,01	16,49
	128	27,91	31,24

Com relação às execuções da versão 6 com a biblioteca OSL, todos os casos com 300 séries apresentaram ganhos de eficiência em relação ao PMO de Março original. Estes ganhos variaram de 1,12%, para o caso com 8 processadores, até 27,91%, para o caso com 128 processadores.

As simulações com a biblioteca COIN apresentaram comportamento similar ao apresentado com a biblioteca OSL. Nas execuções com a versão 5, os casos com 8 e 16 processadores apresentaram pequena perda de eficiência com o aumento de 200 para 300 das séries hidrológicas. A partir do caso com 32 processadores, o aumento da quantidade de séries faz com que a eficiência da estratégia de paralelização melhore, inicialmente com pequenos ganhos, 2,22% para 32 processadores, até atingir 30,77% para 200 processadores. Com relação às execuções com a versão 6, as eficiências com a biblioteca COIN sempre ficaram melhores com o aumento das séries, principalmente com grandes quantidades de processadores.

Outro aspecto importante a destacar neste caso foi o comportamento das eficiências ao longo do processo iterativo para a convergência da solução. A Figura 75 mostra este comportamento ao longo das 16 iterações para as simulações com a versão 5 da biblioteca OSL. Nesta figura, foram mantidas as trajetórias dos casos com 2 e 4 processadores apenas para a comparação com as outras quantidades de processadores. Percebe-se uma clara deterioração da eficiência nos casos com 8, 16 e 32 processadores. Já o caso com 64 processadores também apresentou uma queda na eficiência, porém em escala muito menor que o corrido nos processadores anteriores.

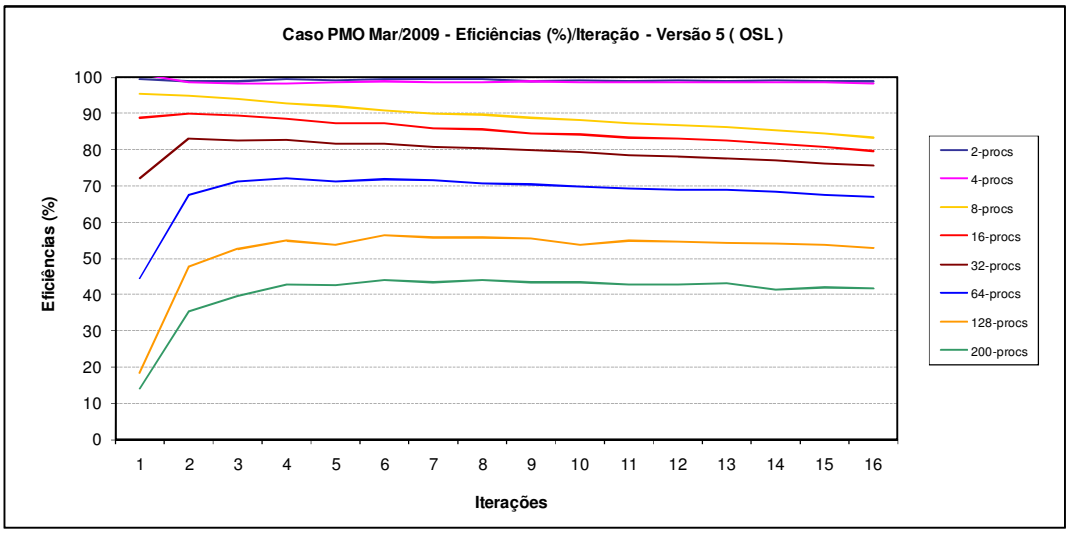

Figura 75 – Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas – Versão 5 – Biblioteca OSL

A execução com a versão 6 e a biblioteca OSL não apresenta mais a deterioração da eficiência ao longo da convergência, conforme está mostrado na Figura 76.

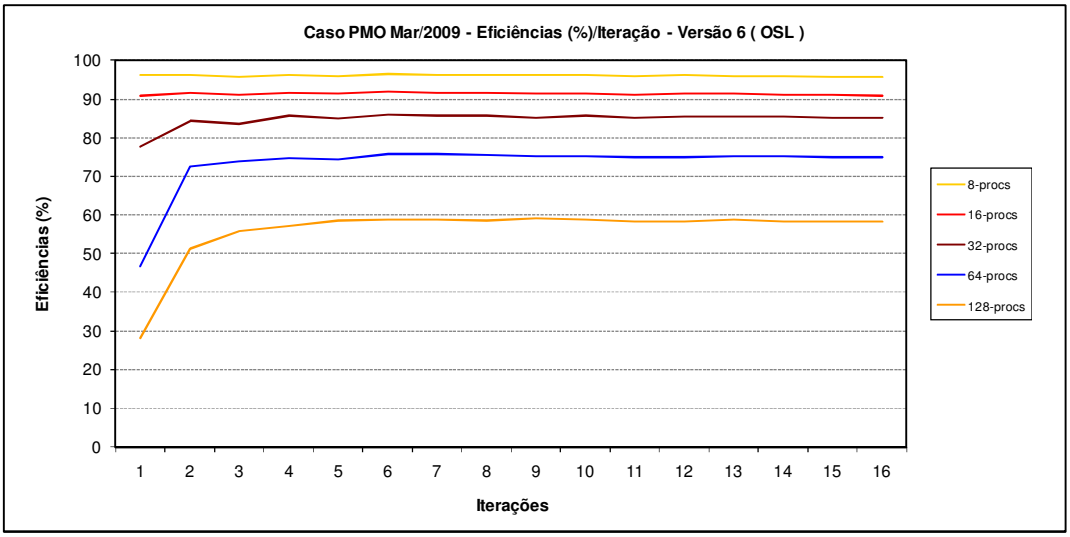

Figura 76 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas – Versão 6 – Biblioteca OSL

Este mesmo evento ocorre nas execuções com as versões 5 e 6 com a biblioteca COIN. Durante a execução da versão 5, apresentado na Figura 77, é claramente visível a queda da eficiência da estratégia de paralelização dos casos até 64 processadores. Inclusive pode-se observar que a queda de desempenho é maior com esta biblioteca do que foi com a OSL. Nas execuções com a versão 6, ainda pode-se observar uma queda

nas eficiências dos casos com 8 e 16 processadores, porém a queda é muito menor do que o ocorrido na execução da versão 5. Com relação às outras quantidades de processadores, a queda de eficiência não foi detectada.

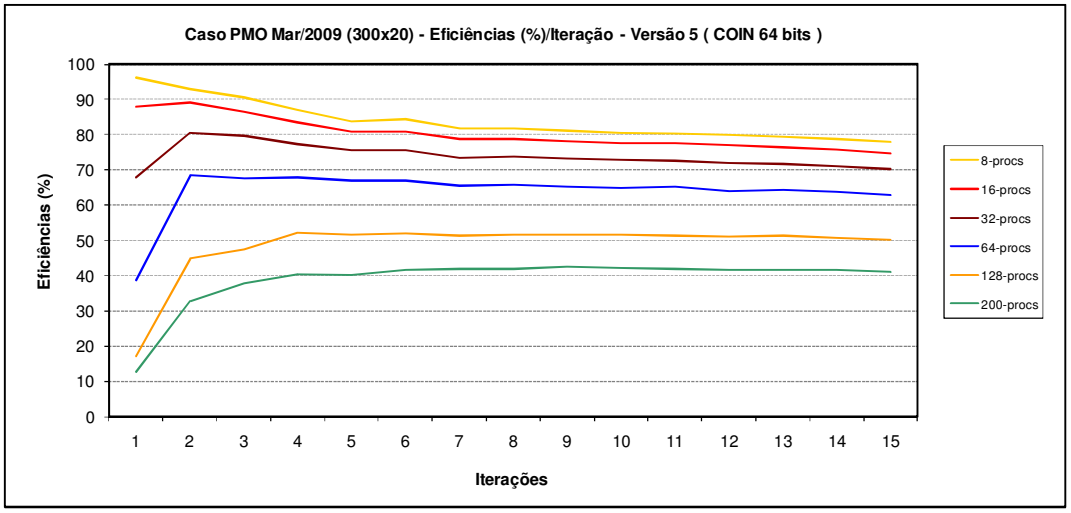

Figura 77 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas – Versão 5 – Biblioteca COIN

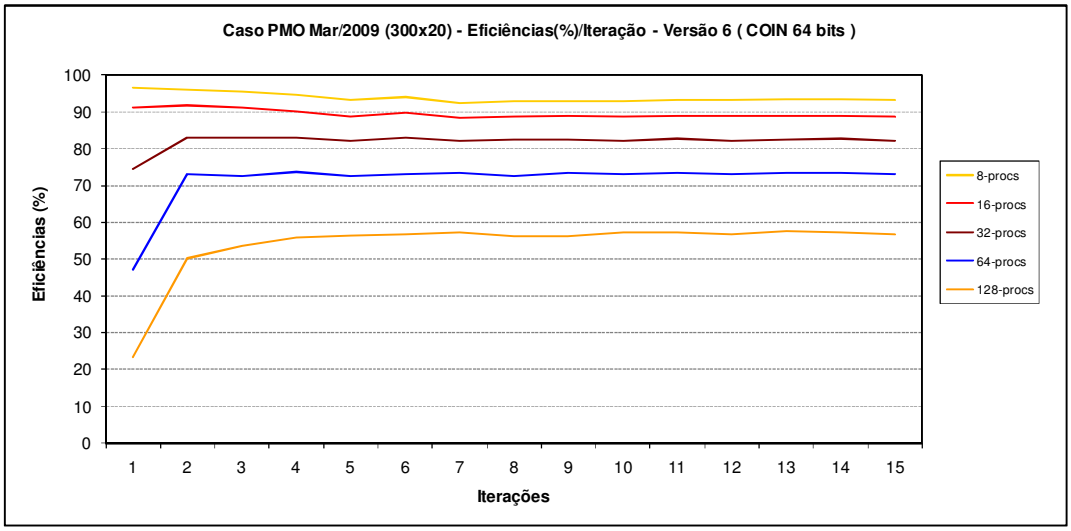

Figura 78 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas – Versão 6 – Biblioteca COIN

6.2.2 Caso PMO de Março de 2009 com 50 Aberturas

O segundo caso com parâmetros aumentados a ser analisado foi com o aumento do número de aberturas de 20 para 50, mantendo a quantidade de séries hidrológicas

constante em 200. Os valores obtidos para as duas bibliotecas de solução de PLs, para as versões 5 e 6, estão mostrados na Tabela 26.

Devem-se ressaltar os tempos necessários para e execução deste caso com 1 processador, que foi de 190h 7min, para as 28 iterações com a biblioteca OSL e 166h 48min para as 29 iterações com a biblioteca COIN. Estes tempos foram bem maiores do que os gastos no problema anterior, principalmente causado pelo processo iterativo bem mais longo para a obtenção das convergências.

Tabela 26 – Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 (200 Séries x 50 Aberturas) para as Opções de Bibliotecas de Solução de PLs OSL e COIN 64 bits

Versão	Qte. Procs.	Biblioteca OSL			Biblioteca COIN 64 bits		
		Tempo (s)	Fator de Aceleração	Eficiência (%)	Tempo (s)	Fator de Aceleração	Eficiência (%)
5	1	684.443	-	-	600.526	-	-
	8	98.768	6,93	86,62	94.772	6,34	79,21
	16	51.504	13,29	83,06	49.066	12,24	76,50
	32	27.945	24,49	76,54	26.449	22,71	70,95
	64	15.859	43,16	67,43	14.911	40,27	62,93
	128	10.329	66,26	51,77	9.354	64,20	50,16
	200	7.606	89,99	44,99	6.811	88,17	44,09
6	8	88.526	7,73	96,64	80.092	7,50	93,72
	16	46.359	14,76	92,27	41.707	14,40	89,99
	32	25.455	26,89	84,03	22.889	26,24	81,99
	64	14.569	46,98	73,41	13.074	45,93	71,77
	128	9.682	70,69	55,23	8.417	71,35	55,74

Com relação à comparação das eficiências deste caso com àquele com os parâmetros originais, na Tabela 27 estão mostradas as diferenças percentuais destes dois casos, tanto com a utilização da versão 5 quanto com a versão 6. Em relação às execuções com a versão 5, percebe-se inicialmente um comportamento similar entre as duas versões com diferentes bibliotecas, tais como, diferenças pequenas para as soluções envolvendo pequenas quantidades de processadores, até os casos com 32 processadores, e diferenças maiores para as demais quantidades de processadores. Para estes casos com maiores quantidades de processadores, a versão com a biblioteca COIN apresentou sempre as maiores diferenças, indicando um maior ganho para um aumento da quantidade de aberturas. As maiores diferenças ocorreram na solução com 200 processadores e foram ambas acima de 40%, praticamente atingindo a marca de 50% para o caso com a biblioteca COIN.

Em relação às execuções com a versão 6, os ganhos de desempenho também ficaram menores nas soluções com menores quantidades de processadores, aumentando

significativamente a partir da solução com 64 processadores. Deve-se ressaltar que, no caso com 128 processadores, a versão com a biblioteca COIN teve um ganho de eficiência bem maior, atingindo uma diferença de 35,6%.

Tabela 27 – Diferenças (%) das Eficiências entre os Casos de PMO de Março de 2009 com 50 e 20 Aberturas

Versão	Qte. Procs.	OSL	COIN 64 bits
5	8	-2,87	-5,02
	16	-0,73	-3,01
	32	1,10	0,21
	64	10,44	12,60
	128	23,20	31,20
	200	41,25	49,52
6	8	1,79	1,53
	16	3,85	3,47
	32	5,73	6,47
	64	13,35	17,37
	128	24,83	35,60

Com relação ao comportamento das eficiências ao longo do processo iterativo, a Figura 79, para a biblioteca OSL, e Figura 80, para a biblioteca COIN, mostram perdas de eficiências, em ambos os casos, maiores do que as encontradas no caso de PMO de Março de 2009 com 300 séries hidrológicas, nas execuções com a versão 5. Outro fato que estas figuras mostram é a perda maior nas simulações com a biblioteca COIN, indicando que o código em 64 bits é mais prejudicado durante a execução com os processadores da família *Core2Quad*.

Quando este caso com 50 aberturas é executado com as versões 6 das bibliotecas OSL e COIN, as eficiências não apresentam mais as quedas anteriormente visualizadas, com os valores se mantendo razoavelmente constantes ao longo do processo iterativo, conforme pode ser visto na Figura 81 e na Figura 82 para as bibliotecas OSL e COIN, respectivamente.

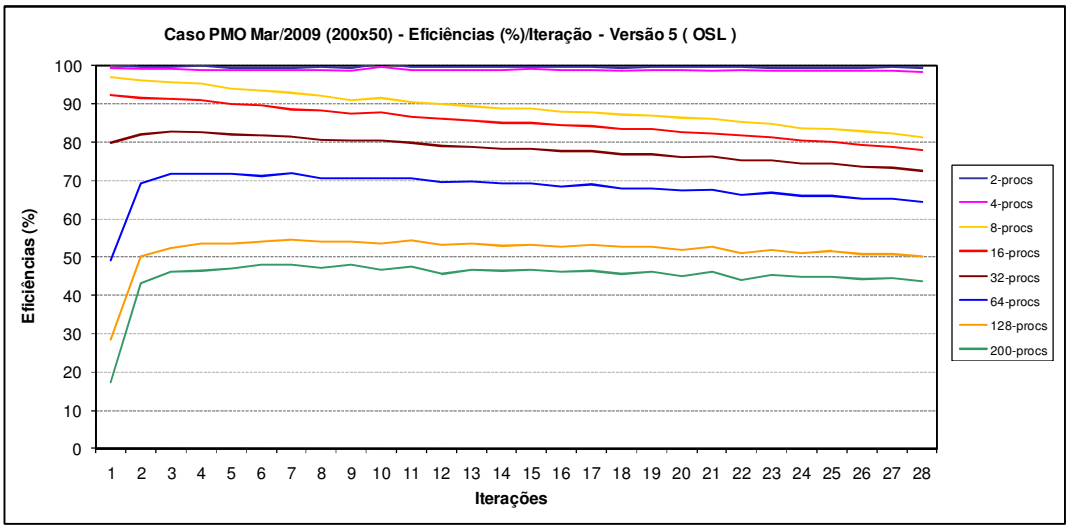

Figura 79 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 50 Aberturas no Ciclo *Backward* – Versão 5 – Biblioteca OSL

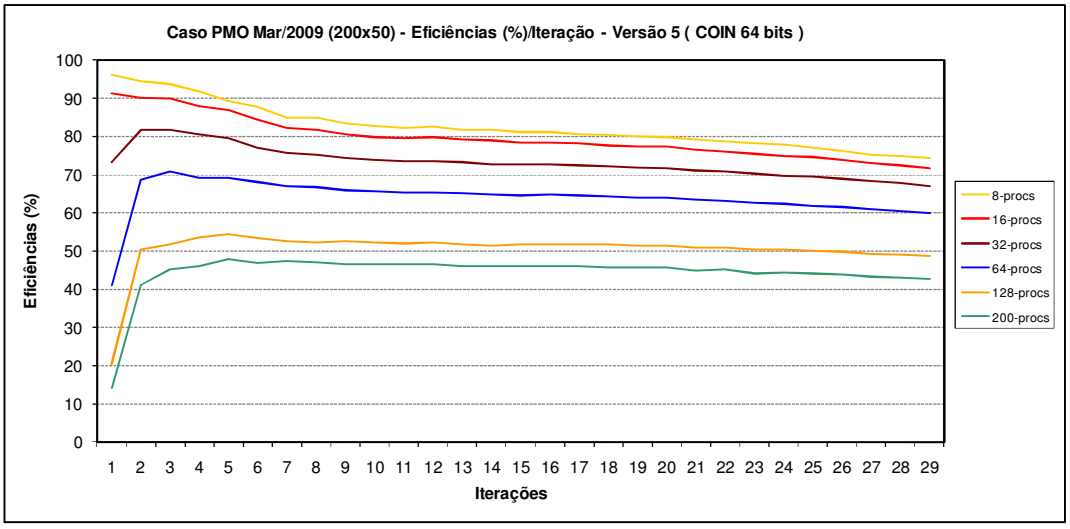

Figura 80 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 50 Aberturas no Ciclo *Backward* – Versão 5 – Biblioteca COIN

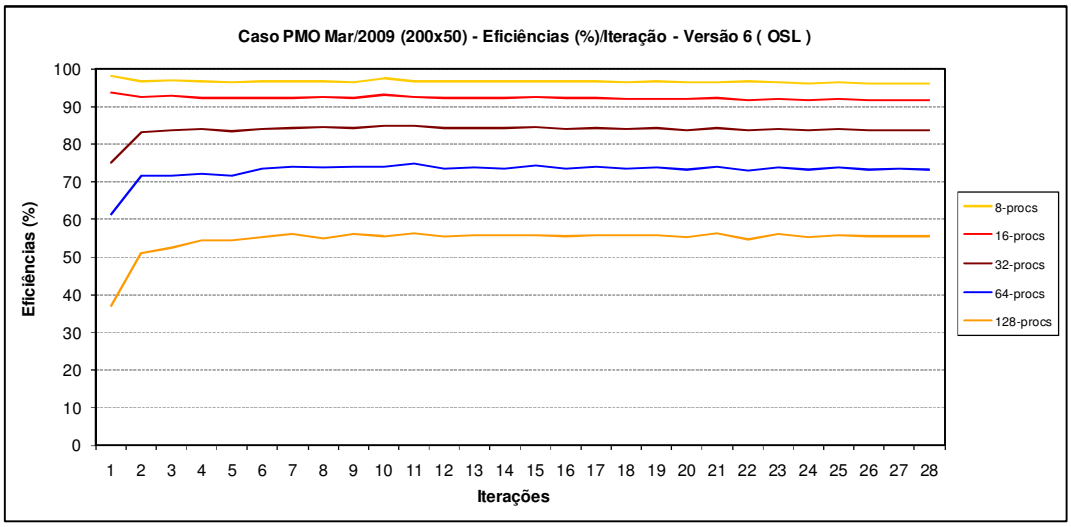

Figura 81 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 50 Aberturas no Ciclo *Backward* – Versão 6 – Biblioteca OSL

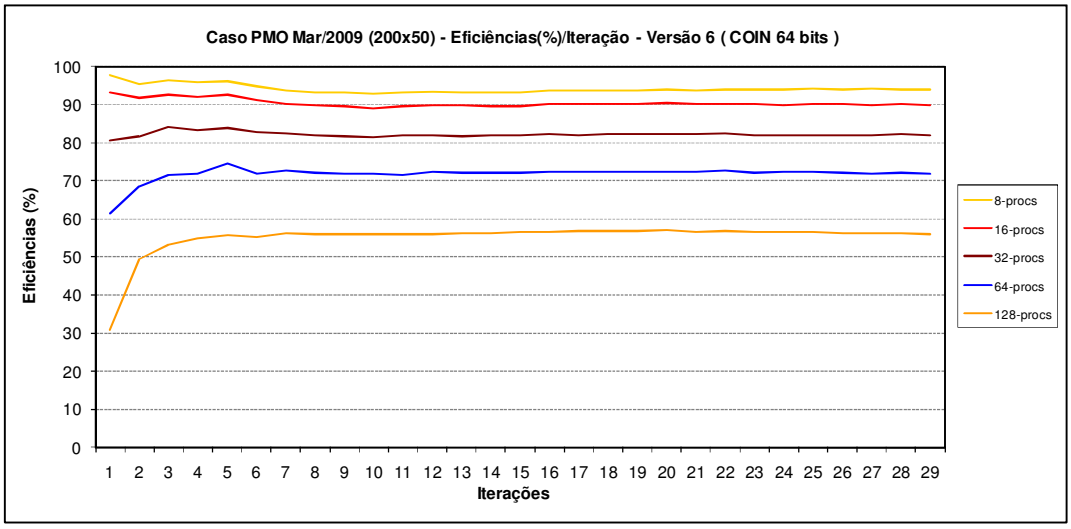

Figura 82 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 50 Aberturas no Ciclo *Backward* – Versão 6 – Biblioteca COIN

6.2.3 Caso PMO de Março de 2009 com 300 Séries e 50 Aberturas

Neste último caso com parâmetros alterados, as quantidades de séries hidrológicas e de aberturas no ciclo *backward* foram colocadas nos limites máximos permitidos pela aplicação. As execuções deste caso com as duas bibliotecas não atingiram a convergência do processo iterativo após 45 iterações, limite imposto pela aplicação, e esses longos processos iterativos fizeram com que os casos levassem tempos exageradamente longos, principalmente nas execuções com poucos

processadores, como por exemplo, 46,5 e 38,5 dias na simulação com 1 processador com as bibliotecas OSL e COIN, respectivamente. Estes tempos muito longos inviabilizam a execução de muitos casos. Por conta disto, decidiu-se executá-lo apenas uma vez para cada quantidade de processador para as duas alternativas de biblioteca. Por outro lado, o processo iterativo excessivamente longo permite uma avaliação da estratégia de paralelização perante uma condição bastante extrema.

Os resultados das execuções estão apresentados na Tabela 28 e pode-se observar que as eficiências dos casos com 8 até 64 processadores, nas execuções com a versão 5, apresentaram eficiências inferiores ao esperado, já que a mesma ficou entre 61,39% e 54,61%, com a biblioteca OSL, e entre 54,22% e 49,30%, com a biblioteca COIN. Os casos com as demais quantidades de processadores também apresentaram eficiências baixas, porém pouco acima do esperado, uma vez que as eficiências destas simulações no caso original tiveram valores parecidos.

Tabela 28 – Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 (300 Séries x 50 Aberturas) para as Opções de Bibliotecas de Solução de PLs OSL e COIN 64 bits

Versão	Qte. Procs.	Biblioteca OSL			Biblioteca COIN 64 bits		
		Tempo (s)	Fator de Aceleração	Eficiência (%)	Tempo (s)	Fator de Aceleração	Eficiência (%)
5	1	4.023.480	-	-	3.331.750	-	-
	8	819.289	4,91	61,39	768.108	4,34	54,22
	16	418.405	9,62	60,10	393.771	8,46	52,88
	32	213.686	18,83	58,84	196.693	16,94	52,93
	64	115.116	34,95	54,61	105.594	31,55	49,30
	128	66.384	60,61	47,35	60.636	54,95	42,93
	200	49.802	80,79	40,39	44.474	74,91	37,46
6	8	518.490	7,76	97,00	440.739	7,56	94,49
	16	273.330	14,72	92,00	229.440	14,52	90,76
	32	147.930	27,20	85,00	122.174	27,27	85,22
	64	79.696	50,49	78,88	67.288	49,51	77,37
	128	48.995	82,12	64,16	41.299	80,67	63,03

Os casos da versão 6 apresentaram eficiências bem mais elevadas, indicando que, neste caso extremo, o problema da deterioração do desempenho do processador em situação de intenso uso de todos os seus núcleos de processamento ficou muito mais evidenciado. Quando, em vez de utilizar os 8 núcleos de processamento disponíveis em cada placa *blade* do *cluster*, são utilizados apenas quatro, os casos com 8 e 16 processadores passam a ter eficiências superiores a 90%. Os casos com 32 processadores apresentaram altas eficiências da ordem de 85% e os casos com 64

processadores ficaram com quase 80% de eficiência. Os casos com 128 processadores apresentaram eficiências de quase 65%, algo que não aconteceu em nenhum dos casos anteriores e que pode ser considerado bastante elevado para esta quantidade de processadores.

Para comparar as eficiências deste caso com as do caso original, foram calculadas as diferenças entre as duas versões, tanto para a biblioteca OSL quanto para a biblioteca COIN, conforme está mostrado na Tabela 29. Para a versão 5 da estratégia de paralelização, as eficiências foram muito mais baixas do que as do caso base, soluções com 8 até 64 processadores, sendo que essas diferenças foram diminuindo com o aumento da quantidade de processadores até que, a partir de 128 processadores, os casos com 300 séries e 50 aberturas passam a ter melhores eficiências.

Tabela 29 – Diferenças (%) das Eficiências entre o Caso com 300 Séries e 50 Aberturas e o Caso Base

Versão	Qte. Procs	OSL	COIN
5	8	-31,17	-34,98
	16	-28,17	-32,95
	32	-22,28	-25,24
	64	-10,76	-11,79
	128	12,69	12,51
	200	26,81	27,03
6	8	2,16	2,36
	16	3,54	4,35
	32	6,95	10,66
	64	21,81	26,53
	128	45,01	53,33

Na comparação com o caso com os parâmetros originais, para a versão 6, os desempenhos foram gradativamente melhor com o aumento da quantidade de processadores, até atingir a maior diferença, na solução com 128 processadores, de 45,01% com a biblioteca OSL (resultado do aumento da eficiência de 44,24% para 64,16%) e 53,33% com a biblioteca COIN (resultado do aumento da eficiência de 41,10% para 63,03%).

Os resultados das eficiências ao longo do processo iterativo, com a versão 5, apresentaram trajetórias bastante descendentes, indicando perdas de desempenho muito elevadas com qualquer uma das duas bibliotecas, conforme pode ser visto na Figura 83, para a biblioteca OSL, e na Figura 84, para a biblioteca COIN. É importante reparar que as trajetórias dos casos com 2 e 4 processadores, na simulação com a biblioteca OSL, não apresentaram a característica de grande perda de eficiência, confirmando que o

problema está associado à utilização de todos os núcleos de processamento dos processadores *Core2Quad*, uma vez que com estas quantidades não são utilizados todos os núcleos de processamento de uma placa *blade*.

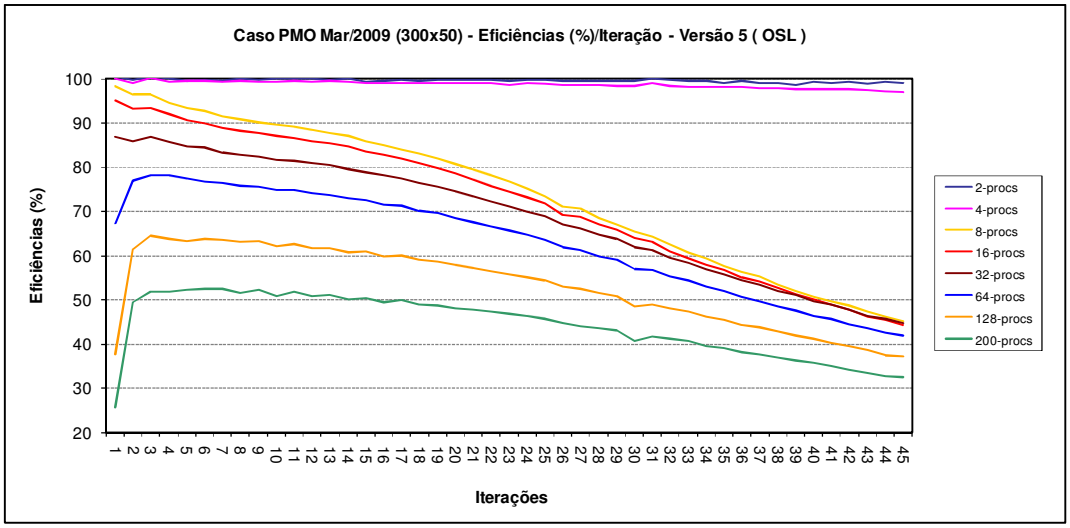

Figura 83 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas e 50 Aberturas – Versão 5 – Biblioteca OSL

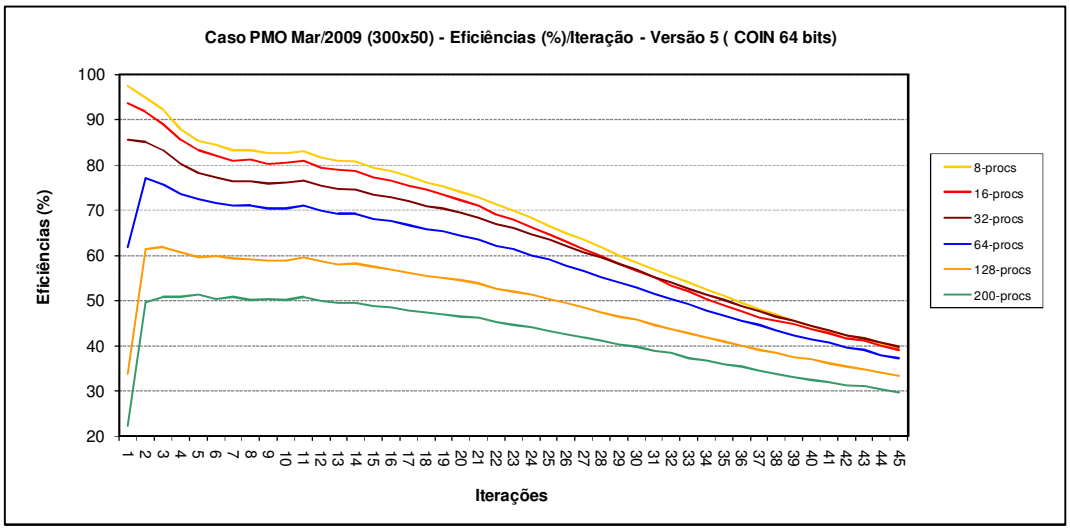

Figura 84 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas e 50 Aberturas – Versão 5 – Biblioteca COIN

As simulações com as versões 6 mostraram uma trajetória de eficiência bastante linear, com perdas insignificantes de desempenho ao longo de todo o processo iterativo, seja com a biblioteca OSL ou com a COIN. Estas eficiências podem ser visualizadas na Figura 85 e na Figura 86 para as bibliotecas OSL e COIN, respectivamente.

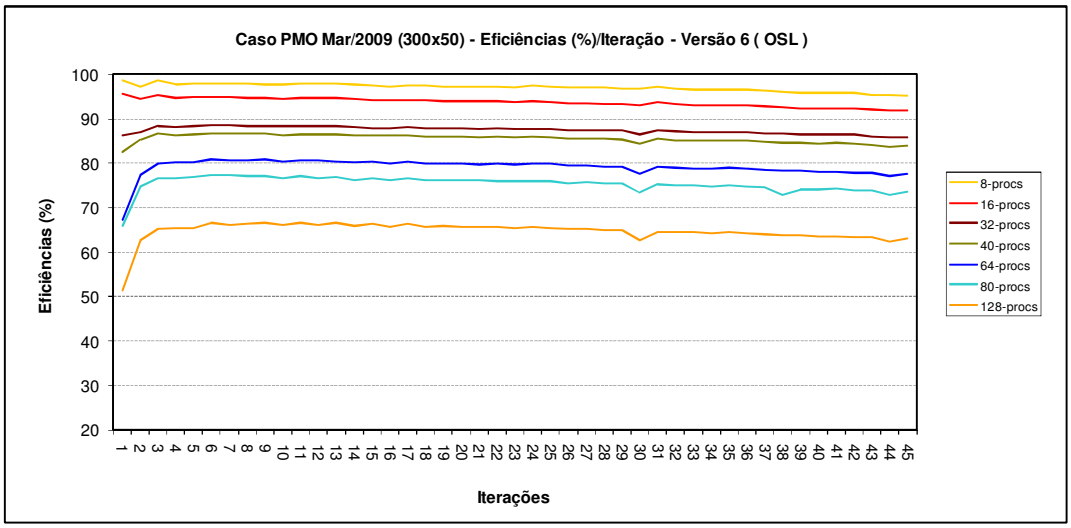

Figura 85 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas e 50 Aberturas – Versão 6 – Biblioteca OSL

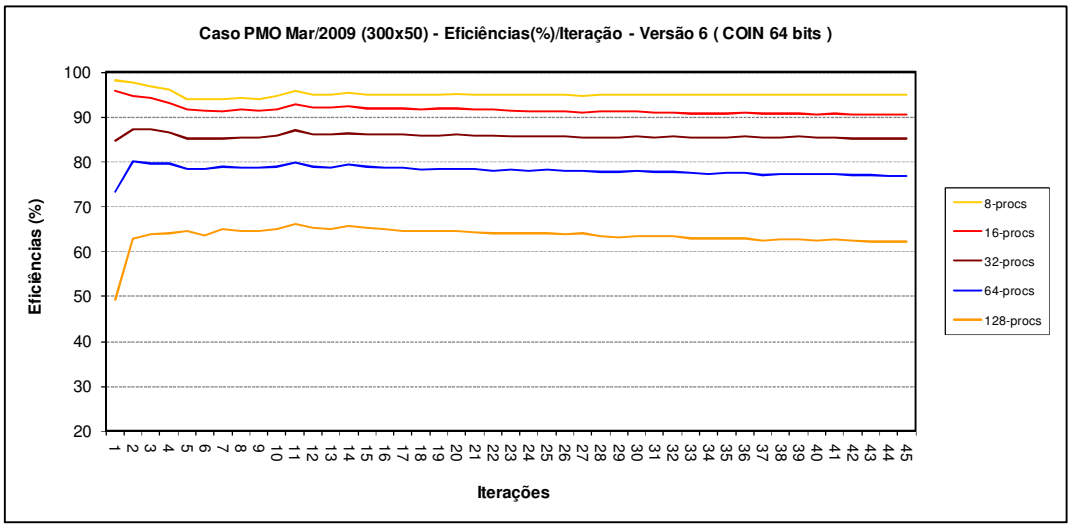

Figura 86 - Eficiências ao Longo das Iterações do Caso PMO de Março de 2009 com 300 Séries Hidrológicas e 50 Aberturas – Versão 6 – Biblioteca COIN

6.2.4 Conclusões

A ampliação do número de séries hidrológicas e das aberturas do ciclo *backward* faz com que os tempos de execução aumentem de forma exponencial. As execuções com a biblioteca OSL, com 1 processador, partiram de 24h 27min, para as 15 iterações do PMO de Março de 2009 original, foram para 60h 45min, para as 16 iterações do caso com 300 séries hidrológicas, aumentaram para 190h 7min, para as 28 iterações do caso

com 50 aberturas e chegaram até a 1.117h 38min (46,5 dias!) para completar as 45 iterações do caso com os aumentos simultâneos para 300 séries hidrológicas e 50 aberturas no ciclo *backward*. A estratégia de paralelização, utilizando 200 processadores, consegue diminuir estes tempos para 23min 2s no caso original, para 44min 50s no caso com 300 séries hidrológicas, para 2h 57min no caso com 50 aberturas e para 13h 36min, com a versão 6 e 128 processadores, no caso com os dois aumentos simultâneos. Estes tempos poderiam ser reduzidos ainda mais, caso fosse possível executar os casos com a versão 6 e 200 processadores, ou até mesmo utilizar 300 processadores para resolver os casos com 300 séries, porém o ambiente onde estão sendo feitas as simulações não possui atualmente recursos computacionais para isso. O uso da estratégia de paralelização produziu reduções significativas de tempo, permitindo obter tempos significativamente menores para os três casos descritos neste item.

As execuções seqüenciais com a biblioteca COIN, apesar de levarem menos tempos do que com a biblioteca OSL, também apresentaram durações bastante elevadas. A execução em 1 processador resultou nos seguintes tempos: 14h 51min para as 13 iterações do caso PMO de Março com os parâmetros originais; 42h 16min para as 15 iterações do caso com 300 séries; 166h 48min para as 29 iterações do caso com 50 aberturas; e 925h 29min (38,5 dias!) para completar as 45 iterações do caso com os aumentos simultâneos para 300 séries hidrológicas e 50 aberturas no ciclo *backward*. As reduções de tempo proporcionadas pela estratégia de paralelização com a biblioteca COIN levaram aos seguintes valores: 15min 7s para o PMO de Março com os parâmetros originais; 32min 53s para o caso com 300 séries; 1h 53min para o caso com 50 aberturas; e 11h 28min para o caso com os aumentos simultâneos para 300 séries hidrológicas e 50 aberturas no ciclo *backward*.

Com relação à comparação das eficiências destes casos, a Figura 87 mostra graficamente as eficiências das execuções da versão 5 com a biblioteca OSL, com 8, 16, 32, 64, 128 e 200 processadores, para as quatro variações do caso PMO de Março de 2009: original com 200 séries hidrológicas e 20 aberturas no ciclo *backward* (200x20), somente ampliação da quantidade de séries hidrológicas para 300 (300x20), somente ampliação da quantidade de abertura de cenários do ciclo *backward* para 50 (200x50) e ampliação simultânea das quantidades de séries hidrológicas e abertura de cenários do ciclo backward para 300 e 50, respectivamente (300x50). Nesta configuração de versão

e biblioteca, a melhor eficiência para os casos com 8 e 16 processadores ocorreu no caso 200x20. Para as quantidades de 32, 64 e 128 processadores, a melhor situação ocorreu no caso 300x20 e para o caso com 200 processadores, a melhor eficiência aconteceu no caso 200x50. Para a maioria das quantidades de processadores, a pior situação aconteceu no caso 300x50, onde as eficiências foram significativamente piores para as quantidades até 64 processadores. Para os casos com 128 e 200 processadores, a pior eficiência ocorreu no caso original (200x20).

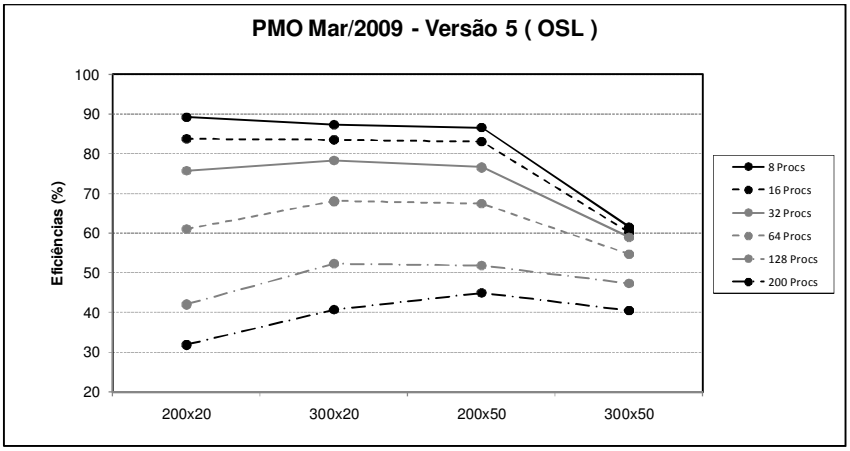

Figura 87 – Eficiências do Caso PMO de Março de 2009 com Variações de Parâmetros – Versão 5 e Biblioteca OSL

Analisando as eficiências da configuração versão 5 e biblioteca COIN, mostrada na Figura 88, observou-se um comportamento bastante similar ao da biblioteca OSL. As melhores e piores eficiências ocorreram nas mesmas situações da versão 5 e biblioteca OSL, só que com valores levemente menores.

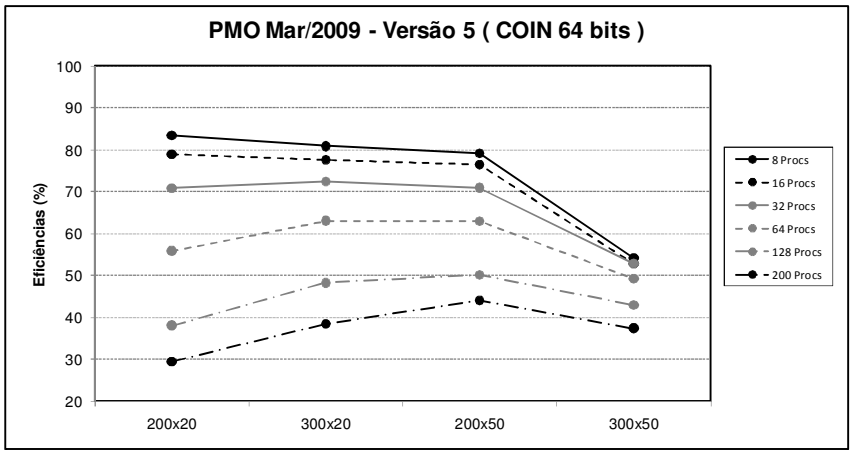

Figura 88 - Eficiências do Caso PMO de Março de 2009 com Variações de Parâmetros – Versão 5 e Biblioteca COIN 64 bits

Com relação às execuções com a versão 6, a Figura 89 e a Figura 90 mostram a evolução das eficiências durante as execuções dos casos analisados neste item, utilizando as bibliotecas OSL e COIN, respectivamente. Pode-se perceber que as simulações com as versões 6 e a biblioteca OSL mostraram que as piores eficiências ocorreram no caso original para todas as quantidades de processadores. Outra característica foi que as eficiências foram sempre ascendentes com a ordem de execução dos casos, ou seja, o caso 300x20 fez com que a estratégia de paralelização ficasse mais eficiente, melhorando um pouco mais no caso 200x50 e atingindo o máximo no caso 300x50. Outro ponto a destacar é o aumento significativo de eficiência dos casos com 64 e 128 processadores na execução com 300 séries e 50 aberturas.

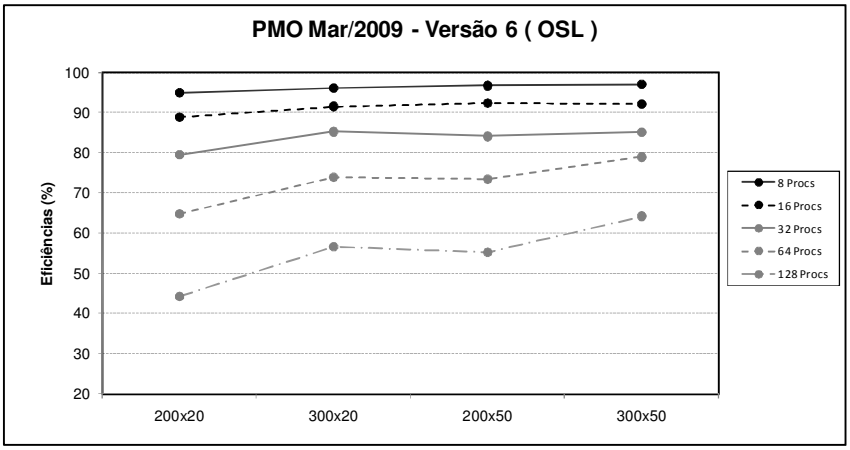

Figura 89 - Eficiências do Caso PMO de Março de 2009 com Variações de Parâmetros – Versão 6 e Biblioteca OSL

O comportamento da versão 6 com a biblioteca COIN mostrou ser bastante similar ao obtido com a biblioteca OSL para os casos com 8, 16 e 32 processadores. Já as simulações com 64 e 128 processadores apresentaram diferenças, uma vez que as eficiências dos casos 300x20 apresentaram eficiências melhores do que os casos 200x50, o que não aconteceu com a biblioteca OSL. Com relação às melhores eficiências, todas ocorreram para o caso 300x50, similarmente ao ocorrido com a outra biblioteca.

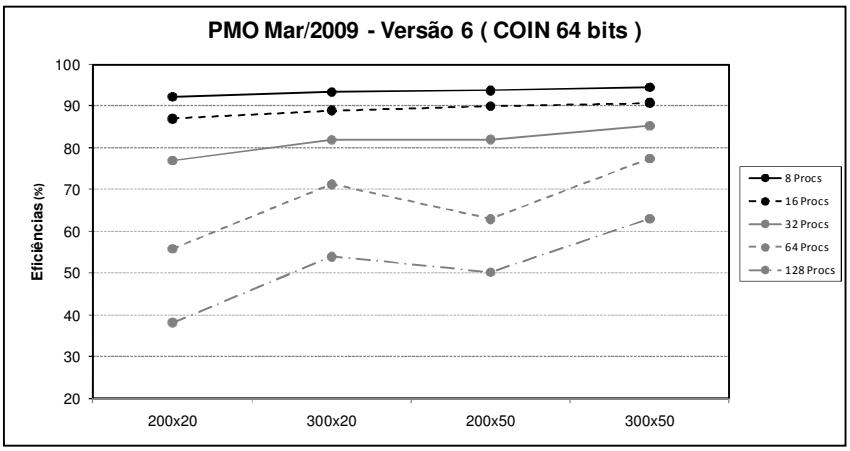

Figura 90 - Eficiências do Caso PMO de Março de 2009 com Variações de Parâmetros – Versão 6 e Biblioteca COIN 64 bits

6.3 Avaliação de Outros Casos de PMO

Os resultados mostrados nos itens anteriores mostraram a coerência dos resultados e também as diferenças de desempenho da estratégia de paralelização proposta entre as diferentes alternativas de bibliotecas de solução de PLs. Porém, é muito importante realizar esta mesma avaliação em outros casos de PMO para que se possam tirar conclusões mais completas, principalmente a cerca do desempenho da estratégia de paralelização. Logo, o objetivo de executar a aplicação com outros casos é verificar o desempenho da estratégia de paralelização, principalmente perante casos que levem mais iterações para serem resolvidos. Estes casos são mais longos e permitem avaliar o comportamento da estratégia em processos iterativos mais longos do que aquele que foi utilizado como base na etapa de otimização da estratégia de paralelização.

Foram escolhidos os casos de PMO de Abril de 2010 e Agosto de 2010, uma vez que estes casos convergiram com mais de 20 iterações. Foram analisadas somente as execuções com as versões 6 da estratégia de paralelização e com as bibliotecas OSL e COIN 64 bits com várias quantidades de processadores. As execuções com as versões 6 fornecem uma expectativa de qual seria a eficiência da estratégia de paralelização se o processador *Core2Quad* não tivesse o problema relatado anteriormente.

6.3.1 Análise das Convergências

Os valores finais dos dois processos de convergência do caso PMO de Abril de 2010 estão apresentados na Tabela 30, onde se pode observar que, apesar das diferentes quantidades de iterações, os valores ficaram bem próximos. Comparando os resultados obtidos pela versão com a biblioteca COIN com os obtidos pela biblioteca OSL, observa-se que a diferença entre os valores de Zinf foi de 0,082%, enquanto que para o Zsup, a diferença foi de 0,042%. Com relação aos limites, tanto os dois valores dos limites máximos quanto dois dos limites mínimos também ficaram bem próximos entre si.

Tabela 30 – Valores Finais das Convergências do Caso PMO Abril de 2010 (Valores em 10^6 R$)

Caso PMO 2010/04					
Biblioteca	Iteração Final	Limite Inferior	Zinf	Limite Superior	Zsup
OSL	23	36.602,70	36.604,49	52.568,36	45.446,36
COIN 64 bits	24	36.605,54	36.634,36	52.497,91	45.427,08

Com relação às trajetórias de convergência, a Figura 91 e a Figura 92 mostram que as trajetórias dos processos iterativos foram bastante parecidas entre si.

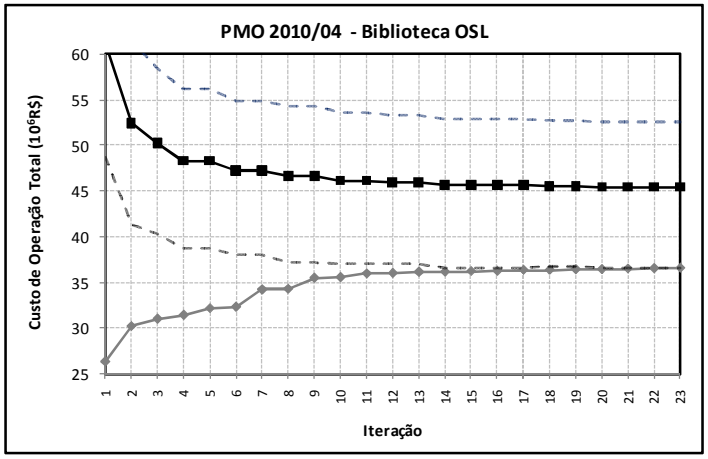

Figura 91 - Processo Iterativo do Caso PMO Abril de 2010 com Biblioteca OSL

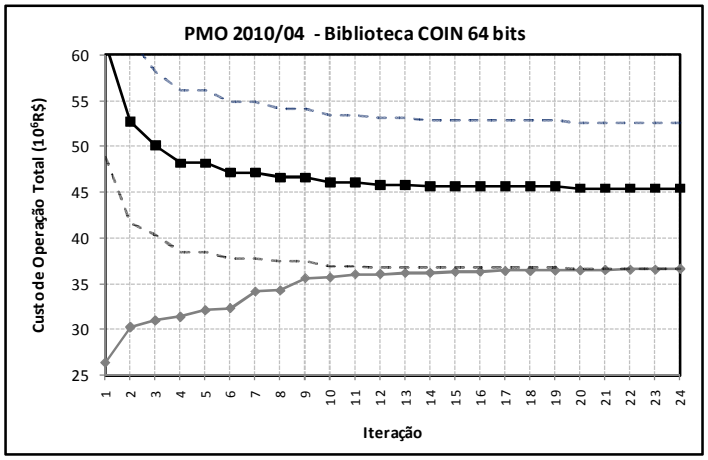

Figura 92 - Processo Iterativo do Caso PMO Abril de 2010 com Biblioteca COIN 64 bits

O segundo caso a ser analisado foi o PMO de Agosto de 2010 e, tal qual o PMO de Abril de 2010, seus resultados finais apresentaram diferenças muito pequenas entre si. Pela Tabela 31, pode-se observar que a diferença entre os valores de Zinf foi de 0,041% e a diferença nos valores de Zsup foi de 0,18%. As diferenças entre os limites máximos e mínimos também ficaram bem pequenas, nunca ultrapassando a marca de 0,29%.

Tabela 31 – Valores Finais das Convergências do Caso PMO Agosto de 2010 (Valores em 10^6 R$)

Caso PMO 2010/08					
Biblioteca	Iteração Final	Limite Inferior	Zinf	Limite Superior	Zsup
OSL	22	40.506,32	40.804,32	58.082,82	50.543,32
COIN 64 bits	21	40.502,96	40.821,25	58.250,91	50.632,48

Para este caso, as trajetórias de convergência das duas bibliotecas analisadas estão mostradas na Figura 93 e na Figura 94, para a OSL e COIN, respectivamente. Apesar dos processos iterativos terem convergido com quantidades de iterações diferentes, pode-se perceber claramente que as trajetórias são bastante similares.

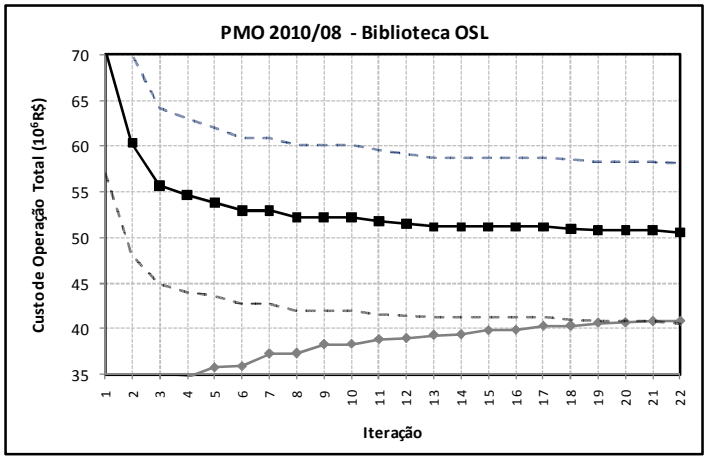

Figura 93 - Processo Iterativo do Caso PMO Agosto de 2010 com Biblioteca OSL

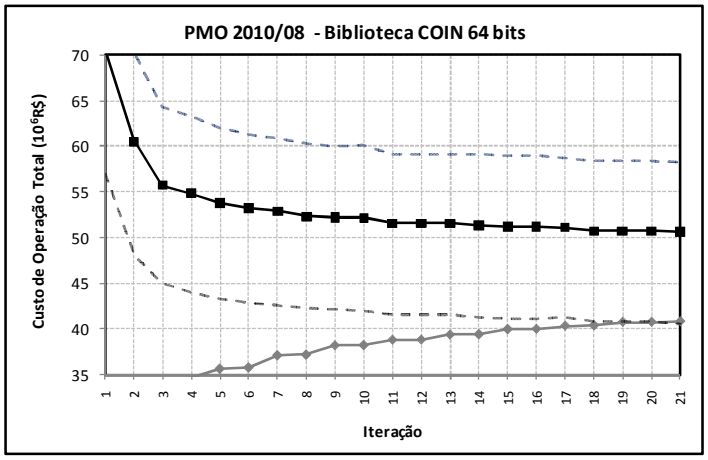

Figura 94 - Processo Iterativo do Caso PMO Agosto de 2010 com Biblioteca COIN 64 bits

Os processos iterativos destes casos mostraram variações na quantidade de iterações para a convergência de cada alternativa de biblioteca. Estas diferenças são devidas a duas razões: (1) a existência de PLs com múltiplas soluções, fazendo com que as soluções adotadas nos problemas pelas bibliotecas possam variar; (2) diferenças nas execuções das operações matemáticas inerentes a forma como as bibliotecas foram programadas e ao processo de compilação, podendo gerar resultados com diferenças

muito pequenas. Com estas duas razões, é perfeitamente possível encontrar soluções diferentes, porém equivalentes em termos estatísticos, entre as duas versões de bibliotecas. Para se ter uma idéia da sensibilidade da convergência, no item 6.1 os códigos fontes das bibliotecas COIN de 32 bits e de 64 bits foram os mesmos, mudando apenas o ambiente em que eles foram compilados, sem mudar a versão do compilador, e os resultados finais foram diferentes. Logo, com versões diferentes de bibliotecas é mais razoável ainda esperar que os resultados não sejam iguais. Porém, os intervalos de confiança das duas soluções precisam mostrar um trecho comum entre eles, indicando a possibilidade da existência de uma mesma solução, o que aconteceu nos casos estudados. É importante destacar também que não existe nenhuma garantia de que a solução com uma biblioteca sempre leve uma quantidade menor, ou maior, de iterações do que a outra. O normal é ocorrer variações neste aspecto, ou seja, algumas soluções encontradas por uma biblioteca levam a um processo iterativo mais rápido em um caso, enquanto que em outro, as escolhas fazem com que o processo iterativo seja mais longo.

6.3.2 Análise das Eficiências

Depois da verificação da coerência dos resultados obtidos pelas duas bibliotecas, é feita a análise das eficiências da estratégia de paralelização para os dois casos de PMO escolhidos. Os quatro conjuntos de resultados dos tempos de execução, fatores de aceleração e eficiências estão mostrados na Tabela 32, para o caso PMO de Abril de 2010, e na Tabela 33 para o caso PMO de Agosto de 2010.

Tabela 32 – Fatores de Aceleração e Eficiências do Caso PMO de Abril de 2010 para as Bibliotecas OSL e COIN 64 bits

Qte. Procs	Biblioteca OSL			Biblioteca COIN 64 bits		
	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	170.759	-	-	159.545	-	-
8	22.600	7,56	94,45	21.701	7,35	91,90
16	12.133	14,07	87,96	11.562	13,80	86,25
32	6.732	25,37	79,27	6.466	24,67	77,10
64	4.100	41,65	65,08	3.921	40,69	63,57
128	2.980	57,30	44,77	2.798	57,03	44,56

Com relação aos tempos de execução, a versão com 1 processador levou 47h 26min com a biblioteca OSL e 44h 19min com a biblioteca COIN 64 bits, um

ganho de 6,5% a favor da biblioteca COIN, mesmo tendo levado uma iteração a mais. Em todas as simulações observou-se que os tempos com a biblioteca COIN foram inferiores, destacando-se os menores tempos médios de 46min 40s com a COIN e 49min 40s com a OSL, ocorridos com 128 processadores.

Com relação às eficiências, a análise dos resultados do caso PMO de Abril de 2010, convergido com 23 e 24 iterações, com as bibliotecas OSL e COIN, respectivamente, mostraram poucas diferenças quando comparadas com as obtidas pelas respectivas versões 6 do caso PMO de Março de 2009, adotado como base neste trabalho.

Nas execuções com a biblioteca OSL, apesar da duração dos casos terem sido por volta do dobro do tempo, as eficiências se mantiveram bastante próximas, um pouco melhores no caso do PMO de Março de 2009 até a quantidade de 32 processadores e pouco melhores no caso do PMO de Abril de 2010 para as quantidades de 64 e 128 processadores. Nos casos com 8, 16 e 32 processadores, as eficiências do PMO de Março de 2009 foram de 94,95%, 88,86% e 79,47%, respectivamente, contra os valores de 94,45%, 87,96% e 79,27% para os respectivos casos com o PMO de Abril de 2010. Para as execuções com 64 e 128 processadores, as eficiências obtidas no PMO de Março de 2009 foram de 64,76% e 44,24%, respectivamente. Já as eficiências destas mesmas quantidades foram respectivamente de 65,08% e 44,77% para o PMO de Abril de 2010.

Em termos de eficiência, o panorama das execuções com a biblioteca COIN foi um pouco diferente, quando se compara as eficiências dos casos de PMO de Março de 2009 e Abril de 2010. As eficiências dos casos com 8 e 16 foram um pouco melhores no PMO de Março de 2009, uma vez que elas foram, respectivamente, de 92,31% e 86,97%, contra 91,90% e 86,25%. Já nos casos com 32, 64 e 128 processadores, as eficiências se mostraram melhores no PMO de Abril de 2010, já que elas foram respectivamente de 77,10%, 63,57% e 44,56%, contra os valores de 77,01%, 61,15% e 41,10%. Pode-se perceber que as diferenças foram um pouco maiores para os casos com 64 e 128 processadores.

O processo de convergência com mais iterações implica na solução de PLs cada vez maiores, e de solução mais lenta, e ainda era desconhecido se este efeito faria com que a eficiência da estratégia de paralelização aumentasse ou diminuísse para casos

originais de meses diferentes. Os resultados mostram que as eficiências ficaram bem próximas entre os dois casos de PMO com as duas bibliotecas, com diferenças um pouco maior para os casos com a biblioteca COIN.

Partindo para a comparação do PMO de Abril de 2010 entre as duas bibliotecas, percebeu-se que as eficiências das execuções foram um pouco melhores nas simulações com a biblioteca OSL, porém os tempos de processamento foram um pouco menores nas simulações com a biblioteca COIN, mesmo com o processo de convergência levando uma iteração a mais.

Da mesma forma como foi feito para o caso de PMO de Março de 2009, os tempos da última iteração comum, que neste caso foi a 23ª, de todas as quantidades de processadores estão apresentados na Tabela 55, localizada no item B.4 do Anexo B. Os resultados mostram uma diferença de pouco mais de 10% nos tempos de execução dos casos com 1 processador em favor da biblioteca COIN. Aliás, as execuções com vários processadores com a biblioteca COIN foram sempre mais rápidas do que com a biblioteca OSL, em média 8,8%, sendo que as diferenças foram crescentes com o aumento da quantidade de processadores utilizadas na solução do caso.

O segundo caso de PMO escolhido foi o de Agosto de 2010, que levou 22 iterações com a biblioteca OSL e 21 iterações com a biblioteca com a biblioteca COIN para convergir. Os resultados com os tempos médios, fatores de aceleração e eficiências estão mostrados na Tabela 33.

Tabela 33 – Fatores de Aceleração e Eficiências do Caso PMO de Agosto de 2010 para as Bibliotecas OSL e COIN 64 bits

Qte. Procs	Biblioteca OSL			Biblioteca COIN 64 bits		
	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	161.347	-	-	121.163	-	-
8	21.306	7,57	94,66	16.476	7,35	91,92
16	11.413	14,14	88,35	8.764	13,83	86,41
32	6.340	25,45	79,53	4.912	24,67	77,09
64	3.852	41,89	65,46	2.975	40,72	63,63
128	2.759	58,48	45,69	2.101	57,67	45,05

Comparando os resultados da biblioteca OSL do PMO de Agosto de 2010 com o de Abril de 2010, percebem-se valores de eficiência muito próximos, porém em todos os casos as eficiências foram ligeiramente superiores no PMO de Agosto. Por exemplo,

na simulação com 8 processadores, a eficiência foi de para 94,66%, no PMO de Agosto, enquanto que no PMO de Abril a eficiência desta simulação foi 94,45%. Este mesmo comportamento também ocorreu com as outras quantidades de processadores.

Com relação à biblioteca COIN, verifica-se também que as eficiências do caso de PMO de Agosto de 2010 foram ligeiramente melhores do que as do caso de PMO de Abril de 2010. Pode-se exemplificar este fato com as eficiências do caso com 8 processadores, que no caso do PMO de Agosto foi de 91,92%, enquanto que o valor obtido no PMO de Abril foi 91,90%. Na prática, pode-se considerar que as eficiências de ambos os casos são iguais e bastante boas.

Numa análise dos tempos de execução, nota-se uma grande diferença nos tempos de execução com 1 processador, que levou 44h 9min com a biblioteca OSL e 33h 39min com a biblioteca COIN, um ganho de tempo de quase 25%, porém deve-se lembrar que o caso com a biblioteca COIN convergiu com uma iteração a menos. Em nenhuma das simulações os tempos das execuções com a biblioteca OSL foram inferiores aos da biblioteca COIN, o que de certa forma é esperado, uma vez que as execuções com a biblioteca COIN têm sido sempre mais rápidas. Deve-se destacar o menor tempo encontrado, que foi no caso com 128 processadores, onde a solução levou praticamente 35 minutos com a biblioteca COIN e 46 minutos com a biblioteca OSL.

6.3.3 Conclusões

Para os três casos de PMO executados com a biblioteca OSL, Março de 2009 (versão 6), Abril e Agosto de 2010, os resultados mostraram que as eficiências das simulações não se alteraram significativamente com a variação da quantidade de iterações para a convergência dos casos. Nos três casos analisados, o processo iterativo levou 15 iterações para convergir o caso PMO de Março de 2009, 22 para o PMO de Agosto de 2010 e 23 para o PMO de Abril de 2010, e o que se pôde constatar foram ligeiras variações de eficiência entre os casos.

Nas simulações com a biblioteca COIN 64 bits, a mesma características de pequena variação entre as eficiências dos casos foi observada, sendo praticamente nula para a maioria do número de processadores.

Outro fato observado foi que as eficiências dos casos com a biblioteca OSL foram superiores aos valores encontrados com a biblioteca COIN, porém as execuções foram sempre mais rápidas nos casos com a biblioteca COIN.

6.4 Desempenho da Estratégia de Paralelização Utilizando Processadores de Última Geração

Na etapa para a geração da versão 5, foi observada uma queda gradual da eficiência da estratégia de paralelização para os casos executados a partir de oito processadores. Observou-se que a queda da eficiência ou era bastante reduzida, ou era eliminada, quando se reservava o dobro da capacidade de processamento do caso, o que levou a criação da versão 6. A partir da análise dos tempos de processamento das tarefas da rotina *backward*, chegou-se a conclusão que o problema deveria ser do processador propriamente dito. Logo, a única maneira de confirmar as suspeitas seria a execução dos mesmos casos numa máquina que tivesse seus processadores de uma família mais recente cuja arquitetura interna fosse diferente daquela do processador utilizado no *cluster* do CEPEL.

Diante da necessidade encontrada, o Núcleo de Atendimento em Computação de Alto Desempenho (NACAD) da COPPE/UFRJ disponibilizou um servidor com 2 processadores Xeon da família *Nehalem*, pertencentes à última geração de processadores da Intel. Cada processador deste servidor possui quatro núcleos, totalizando oito núcleos de processamento disponíveis para a execução dos casos. Para realizar uma avaliação completa, foram executados três casos referentes ao PMO de Março de 2009, um com os parâmetros originais (200 séries e 20 aberturas – 200x20) e os outros dois com os parâmetros aumentados, um com 300 séries (300x20) e outro com 50 aberturas (200x50). Estes casos foram executados com a versão 5 da estratégia de paralisação utilizando as bibliotecas OSL e COIN 64 bits.

6.4.1 Caso PMO de Março de 2009

Neste item serão analisados os resultados obtidos com as bibliotecas OSL e COIN para o caso PMO de Março de 2009, com os parâmetros originais. Os resultados para as soluções com 1 e 8 processadores estão mostrados na Tabela 34.

Tabela 34 - Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 com o Processador Intel Xeon *Nehalem*

Qte. Procs	Biblioteca OSL			Biblioteca COIN		
	Tempo (s)	Fator de Aceleração	Eficiência (%)	Tempo (s)	Fator de Aceleração	Eficiência (%)
1	73.211	-	-	41.775		
8	10.030	7,30	91,24	5.719	7,30	91,31

- **Biblioteca OSL**

Com relação aos resultados da versão com a biblioteca OSL, comparando estes resultados com os apresentados na Tabela 17, para a versão 5, e na Tabela 19, para a versão 6, pode-se observar que a eficiência do caso com 8 processadores (91,24%) situou-se entre os valores dos dois casos executados no processador *Core2Quad* (89,18% e 94,95%). Apesar da eficiência do caso executado no processador Intel *Nehalem* ter sido um pouco pior do que a melhor eficiência obtida com o processador *Core2Quad,* os tempos finais neste processador foram significativamente menores, tanto no caso com um processador quanto no caso com oito processadores, conforme pode ser visto na Tabela 35. Pode-se observar, por exemplo, que a simulação com 1 processador no processador *Nehalem* foi 16,85% mais rápida, enquanto que a simulação com 8 processadores foi 13,47% mais rápida do que a execução com a versão 6 no processador *Core2Quad*.

Tabela 35 – Diferenças entre as Execuções nos Processadores *Core2Quad* e *Nehalem* com a Biblioteca OSL (Caso PMO Março 2009)

Qte. Procs	Processador *Core2Quad* Versão 5 (s)	Processador *Core2Quad* Versão 6 (s)	Processador *Nehalem* (s)	Diferença *Nehalem* *Core2Quad* Versão 5 (%)	Diferença *Nehalem* *Core2Quad* Versão 6 (%)
1	88.045	-	73.211	16,85	-
8	12.341	11.591	10.030	18,73	13,47

Para descobrir se a execução no processador *Nehalem* possui a mesma característica de perda de desempenho ao longo do processo de convergência, os valores das eficiências de cada iteração foram obtidos e comparados com os do mesmo caso no processador *Core2Quad*, para as versões 5 e 6, conforme está apresentado na Figura 95.

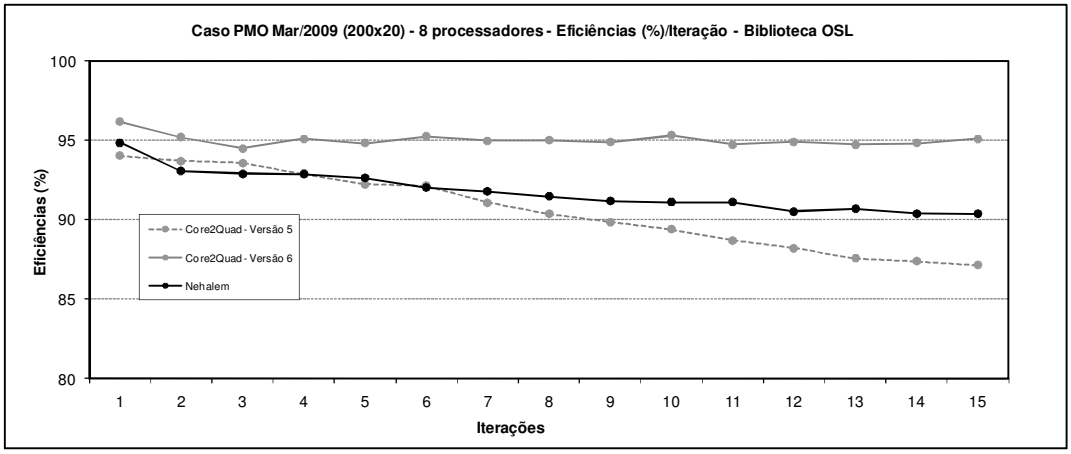

Figura 95 - Comparação das Eficiências para 8 Processadores
(Caso PMO Março/2009 - 200x20- OSL)

Observa-se claramente uma perda de desempenho ao longo do processo iterativo do caso executado no processador *Nehalem*, porém ela estabiliza antes e num patamar superior ao que ocorreu na versão 5 executada no processador *Core2Quad*. Uma possível explicação pode ser o desempenho do disco onde os dados são lidos e escritos, uma vez que o disco utilizado no cluster do CEPEL é do padrão SAS, mais rápido do que o padrão SATA, que é utilizado no servidor disponibilizado no NACAD. Com um disco mais lento, as operações que dependem do acesso a este disco tendem a consumir mais tempo, fazendo com que o desempenho fique um pouco pior.

- **Biblioteca COIN 64 bits**

Com relação aos resultados com a biblioteca COIN 64 bits, a eficiência de 91,31% (Tabela 34) para a solução com 8 processadores também se situou entre as execuções desta mesma biblioteca com as versões 5 e 6, 83,40% e 92,31%, respectivamente (Tabela 22), no processador *Core2Quad*. O comportamento com esta biblioteca diferiu um pouco do apresentado pela biblioteca OSL, uma vez que foi muito mais próximo da eficiência da versão 6 do que do apresentado na versão 5. Tal qual ocorrido com a biblioteca OSL, a execução com a biblioteca COIN no processador *Nehalem* foi significantemente mais rápida que o mesmo caso executado no *cluster* com processadores *Core2Quad*, tendo os ganhos de tempo situados entre 21-29%, conforme pode ser visto na Tabela 36.

Tabela 36 – Diferenças das Execuções nos Processadores *Core2Quad* e *Nehalem* com a Biblioteca COIN 64 bits (Caso PMO Março 2009)

Qte. Procs	Processador *Core2Quad* Versão 5 (s)	Processador *Core2Quad* Versão 6 (s)	Processador *Nehalem* (s)	Diferença *Nehalem Core2Quad* Versão 5 (%)	Diferença *Nehalem Core2Quad* Versão 6 (%)
1	53.509	-	41.775	21,93	
8	8.027	7.249	5.719	28,75	21,11

O outro ponto a ser analisado é a verificação do desempenho do código 64 bits neste novo processador ao longo do processo iterativo. Com este objetivo, os tempos totais de cada iteração para a execução no processador *Nehalem* e as execuções das versões 5 e 6 no processador *Core2Quad* foram traçados, conforme está mostrado na Figura 96. Pela figura percebe-se que a execução no servidor do NACAD permitiu que a eficiência da estratégia de paralelização se mantivesse com pouca variação a partir da segunda iteração, fazendo com que a sua eficiência ficasse muito próxima do obtido pela execução da versão 6. A execução no processador *Nehalem*, para este caso, não apresentou a forte queda observada na versão 5 com o processador *Core2Quad*. Ou seja, a execução da versão 5 no processador *Nehalem* é pelo menos 20% mais rápido e tem praticamente a mesma eficiência da execução da versão 6 no processador *Core2Quad*. Estes resultados mostram a grande diferença entre os processadores de famílias diferentes, quando executados com o código de 64 bits.

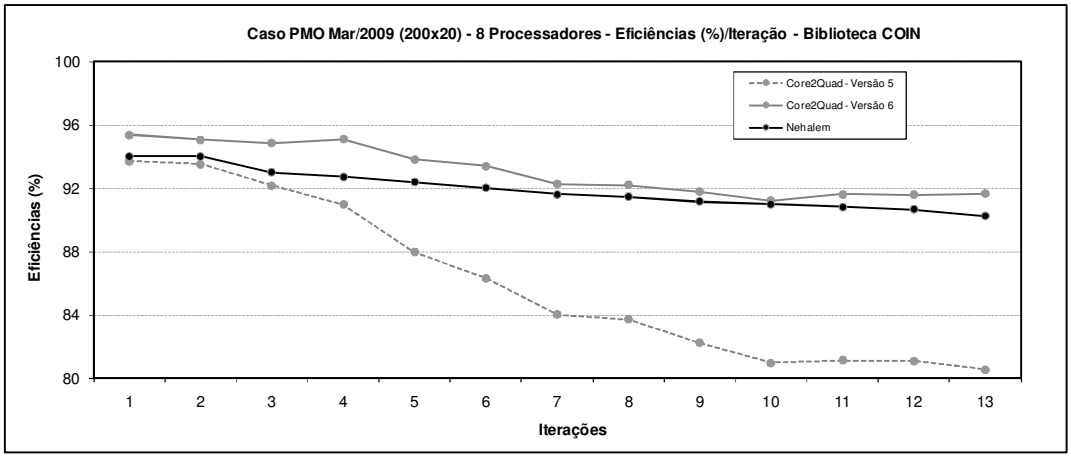

Figura 96 - Comparação das Eficiências para 8 Processadores (Caso PMO Março/2009 - 200x20- COIN 64 bits)

6.4.2 Caso PMO de Março de 2009 com 300 Séries

O próximo caso a ser analisado foi a expansão da quantidade de séries hidrológicas do caso PMO de Março de 2009 de 200 para 300, mantendo a quantidade de aberturas do ciclo *backward* em 20 (300x20). Os valores das eficiências para os casos com 8 processadores para as duas bibliotecas de solução de PLs estão mostrados na Tabela 37.

Tabela 37 – Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 (300 Séries) para as Bibliotecas de Solução de PLs com Processador Intel Xeon *Nehalem*

Qte. Procs	Biblioteca OSL			Biblioteca COIN 64 bits		
	Tempo (s)	Fator de Aceleração	Eficiência (%)	Tempo (s)	Fator de Aceleração	Eficiência (%)
1	177.693	-	-	114.934	-	-
8	24.517	7,25	90,60	15.704	7,32	91,48

- **Biblioteca OSL**

Com relação à eficiência deste caso, o valor de 90,60% também ficou entre os valores das versões 5 e 6, 87,30% e 96,02% (Tabela 24), respectivamente. Porém, diferentemente do que ocorreu no caso original, o valor ficou mais próximo do limite inferior do que do superior. Também é interessante notar que a eficiência deste caso foi menor do que ocorreu no caso com os parâmetros originais, em que a eficiência foi de 91,24%. Este comportamento foi o inverso do ocorrido entre estes dois casos quando foram executados no *cluster* do CEPEL, onde a eficiência do caso com 300 séries (96,02%) foi maior do que o caso com 200 séries (93,16%).

Com relação à comparação dos tempos de processamento, os resultados deste caso estão mostrados na Tabela 38, onde se podem observar os ganhos de tempo obtidos na execução com o processador *Nehalem* , que ficou entre 14-22%.

Tabela 38 – Diferenças entre as Execuções nos Processadores *Core2Quad* e *Nehalem* com a Biblioteca OSL (Caso PMO Março 2009 com 300 Séries)

Qte. Procs	Processador *Core2Quad* Versão 5 (s)	Processador *Core2Quad* Versão 6 (s)	Processador *Nehalem* (s)	Diferença *Nehalem* *Core2Quad* Versão 5 (%)	Diferença *Nehalem* *Core2Quad* Versão 6 (%)
1	218.717	-	177.693	18,76	-
8	31.318	28.474	24.517	21,72	13,90

Na Figura 97 estão apresentadas as características das eficiências ao longo do processo iterativo dos três casos com 8 processadores. Pode-se observar que a eficiência

do caso executado com o processador *Nehalem* apresentou uma queda de 96% para 92%, mantendo este valor até quase o final do processo iterativo. Comparando com o da versão 5 no processador *Core2Quad*, nota-se que apenas após a 10ª iteração é que a eficiência no processador *Nehalem* fica com um valor superior àquele.

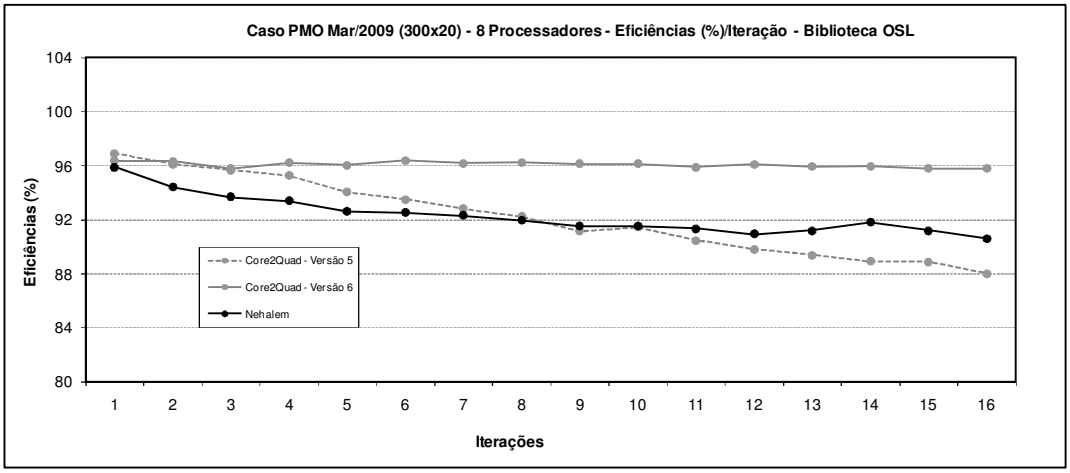

Figura 97 - Comparação das Eficiências para 8 Processadores
(Caso PMO Março/2009 - 300x20- OSL)

- **Biblioteca COIN**

A eficiência do caso com a biblioteca COIN foi de 91,48%, situando-se também entre as eficiências de 80,87% da versão 5 e 93,34% da versão 6 com o processador *Core2Quad* (Tabela 24). Já com relação aos tempos de execução, a Tabela 39 mostra ganhos entre 23% e 33%, mostrando que, proporcionalmente, esta versão apresentou ganhos maiores do que os apresentados pela biblioteca OSL para este caso.

Tabela 39 – Diferenças das Execuções nos Processadores *Core2Quad* e *Nehalem* com a Biblioteca COIN 64bits (Caso PMO Março 2009 com 300 Séries)

Qte. Procs	Processador *Core2Quad* Versão 5 (s)	Processador *Core2Quad* Versão 6 (s)	Processador *Nehalem* (s)	Diferença *Nehalem Core2Quad* Versão 5 (%)	Diferença *Nehalem Core2Quad* Versão 6 (%)
1	152.186	-	114.934	24,48	-
8	23.524	20.380	15.704	33,24	22,94

Para se avaliar a trajetória da eficiência ao longo do processo de convergência deste caso, na Figura 98 estão mostradas as eficiências das execuções com os processadores *Nehalem* e *Core2Quad*. Pode-se perceber que a trajetória das eficiências deste caso possui uma lenta deterioração, partindo de 95% e terminando pouco acima de

90%. A execução desta versão de 64 bits num processador da família *Nehalem* não apresentou a grande deterioração da eficiência da versão 5 executada no processador *Core2Quad*, se mantendo em valores próximos dos da versão 6.

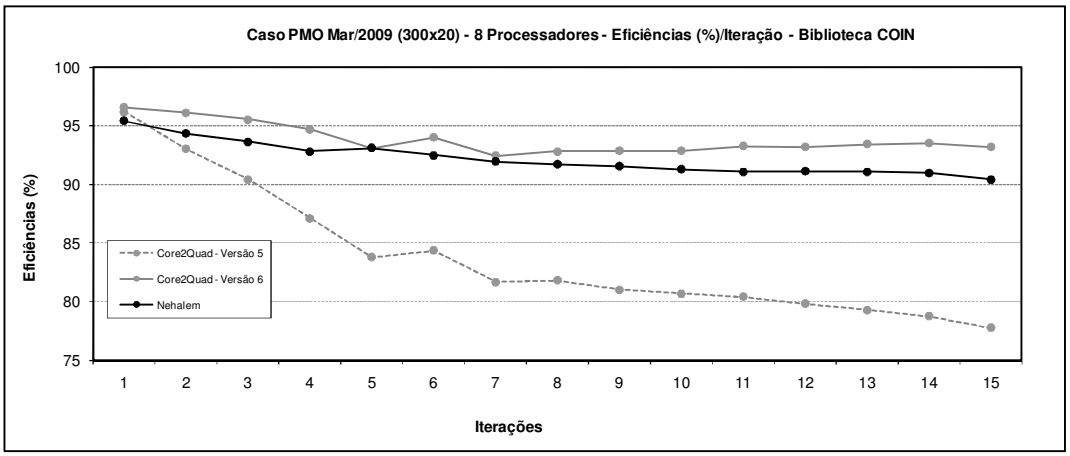

Figura 98 - Comparação das Eficiências para 8 Processadores
(Caso PMO Março/2009 - 300x20- COIN 64 bits)

6.4.3 Caso PMO de Março de 2009 com 50 Aberturas

Neste caso, aumentou-se a quantidade de aberturas do caso PMO de Março de 2009 de 20 para 50 e manteve-se a quantidade de séries hidrológicas em 200 (200x50). Na Tabela 40, estão apresentados os resultados de tempo de execução, fatores de aceleração e eficiências para as duas opções de bibliotecas adotadas neste trabalho.

Tabela 40 – Fatores de Aceleração e Eficiências do Caso PMO de Março de 2009 (50 Aberturas) para as Bibliotecas de Solução de PLs com Processador Intel Xeon *Nehalem*

Qte. Procs	Biblioteca OSL			Biblioteca COIN 64 bits		
	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)	Tempo Médio (s)	Fator de Aceleração	Eficiência (%)
1	562.049	-	-	448.456	-	-
8	76.773	7,32	91,51	61.444	7,30	91,23

Os resultados mostram que o caso com a biblioteca OSL foi lento, resolvendo o caso em 156h 7min (562.049s), enquanto que este mesmo caso foi resolvido em 124h 34min (448.456s), 20% mais rápido, mesmo levando uma iteração a mais para convergir o processo iterativo.

- **Biblioteca OSL**

Este caso executado com a biblioteca OSL apresentou uma eficiência de 91,51%, também entre as eficiências das versões 5 e 6, que ficaram em 86,62% e 96,64%, respectivamente (Tabela 26).

A Tabela 41 mostra as diferenças de tempo entre as execuções com 1 e 8 processadores. Percebe-se que a execução com o novo processador sempre apresentou redução de tempo, mesmo quando comparado com a versão 6, onde é reservado o dobro da capacidade de processamento no processador mais antigo. Os ganhos de tempo da execução com o *Nehalem* foram bastante parecidos com os apresentados nos outros casos com esta biblioteca, ficando as reduções de tempo entre 13 e 14% em comparação com a versão 6 no processador *Core2Quad*. Na execução seqüencial (com 1 processador), as reduções foram bem maiores, ficando em 16,78%, no caso 200x20, em 18,76%, no caso 300x20, e 17,88% no caso 200x50. Este ganho maior para os casos com 1 processador, em relação ao ganho com 8 processadores, implica em eficiências menores nestes casos com o servidor do NACAD em relação à execução com a versão 6 no *cluster* do CEPEL.

Tabela 41 – Diferenças entre as Execuções nos Processadores *Core2Quad* e *Nehalem* com a Biblioteca OSL (Caso PMO Março 2009 com 50 Aberturas)

Qte. Procs	Processador *Core2Quad* Versão 5 (s)	Processador *Core2Quad* Versão 6 (s)	Processador *Nehalem* (s)	Diferença *Nehalem Core2Quad* Versão 5 (%)	Diferença *Nehalem Core2Quad* Versão 6 (%)
1	684.443	-	562.049	17,88	-
8	98.768	88.526	76.773	22,27	13,28

Para verificar como foi o comportamento da eficiência ao longo dos processos iterativos, foi traçado o gráfico mostrado na Figura 99, onde as eficiências das execuções com 8 processadores estão apresentadas. A eficiência do processador *Nehalem* começa com um valor em torno de 96%, possui uma queda gradual até 92% na 8ª iteração, mantendo-se neste patamar até aparecer um pico de quase 96% da 16ª iteração e depois ficando por volta de 91% com uma pequena deterioração até o final do processo iterativo.

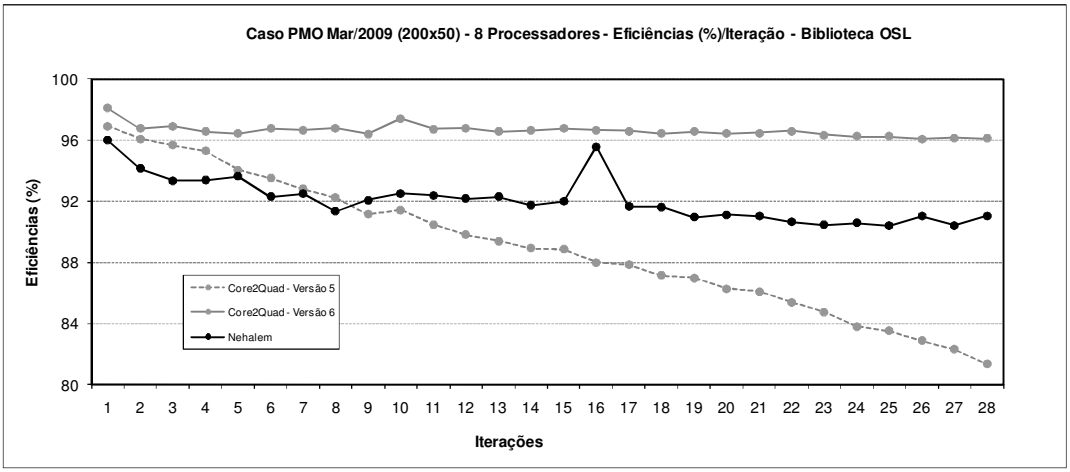

Figura 99 - Comparação das Eficiências para 8 Processadores
(Caso PMO Março/2009 - 200x50 - OSL)

- **Biblioteca COIN**

A execução do caso com a biblioteca COIN apresentou uma eficiência de 91,23%, praticamente igual à eficiência de 91,27% do caso 200x20, indicando que o aumento deste parâmetro não resultou em melhora na eficiência da estratégia de paralelização. Este valor situa-se entre os obtidos quando o caso é executado no processador *Core2Quad* (Tabela 26), onde as eficiências foram de 79,21% com a versão 5 e 93,72% com a versão 6. Apesar da eficiência deste caso no processador *Nehalem* não ter atingido a eficiência obtida com a versão 6, ela foi próxima, com a vantagem de se utilizar somente os recursos computacionais necessários para a execução do caso. Outro aspecto importante é que a execução do caso foi bem mais rápida, levando entre 23% e 35% menos tempo do que na execução no processador *Core2Quad* (Tabela 42).

Tabela 42 – Diferenças das Execuções nos Processadores *Core2Quad* e *Nehalem* com a Biblioteca COIN 64bits (Caso PMO Março 2009 com 50 Aberturas)

Qte. Procs	Processador *Core2Quad* Versão 5(s)	Processador *Core2Quad* Versão 6 (s)	Processador *Nehalem* (s)	Diferença *Nehalem* *Core2Quad* Versão 5 (%)	Diferença *Nehalem* *Core2Quad* Versão 6 (5)
1	600.526	-	448.456	25,32	-
8	94.772	80.092	61.444	35,17	23,28

Para verificar o comportamento da eficiência da estratégia de paralelização ao longo do processo de convergência, a Figura 100 mostra que a mesma apresentou um processo lento e gradual de queda até a 17ª iteração, a partir da qual se manteve num nível pouco acima de 90%. Este mesmo processo na versão 6 situou-se num nível final

mais elevado, por volta de 94%, além do processo de queda ter parado a partir da 10ª iteração.

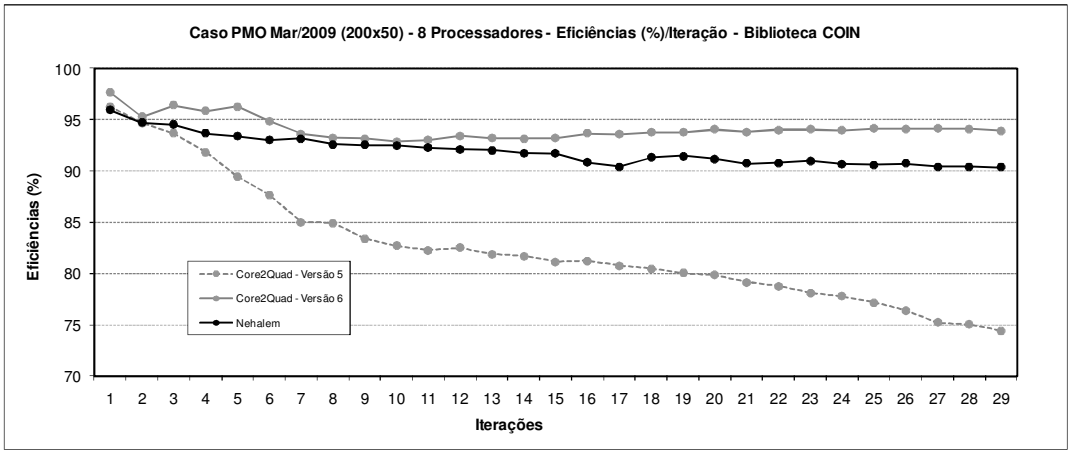

Figura 100 - Comparação das Eficiências para 8 Processadores
(Caso PMO Março/2009 - 200x50 - COIN 64 bits)

O que se pode reparar destes resultados é que o processo de perda de eficiência do processador da família mais antiga é mais crítico quando o processo iterativo aumenta e quando se usa a configuração de 64 bits. Logo, a vantagem do uso de um processador de concepção mais moderna será mais significativa quanto mais longo for o processo de convergência e com o uso de versões de 64 bits, onde as diferenças entre os tempos de execução dos cálculos ficaram ainda maiores.

6.4.4 Conclusões

Os resultados obtidos com o servidor do NACAD foram muito interessantes. A primeira constatação é que as eficiências foram bastante parecidas nos três casos executados, tanto para a biblioteca OSL quanto para a biblioteca COIN. Estas eficiências variaram de 90,60%, biblioteca OSL e caso com 300x20, até 91,51%, biblioteca OSL e caso 200x50. Em dois casos, a eficiência da estratégia de paralelização com a biblioteca COIN foi levemente melhor do que com a biblioteca OSL. Estes casos foram o 200x20, 91,31% contra 91,24% com a OSL, e 300x20, 91,48% contra 90,60% com a OSL. Ou seja, a execução em processadores de concepção mais moderna fez com a eficiência da estratégia de paralelização fosse mais homogênea em relação à variação de caso e de biblioteca de solução de PLs, porém, deve-se ressaltar que os tempos com a

biblioteca COIN foram significativamente mais rápidos que com a OSL. Além disso, é importante registrar que esta conclusão se refere apenas à eficiência da simulação com 8 processadores, devendo ser confirmada no futuro com as demais quantidades de processadores.

Com relação ao ganho de tempo ocorrido nas simulações no processador *Nehalem*, eles foram, em média, de 13,6% com a biblioteca OSL e de 22,4% com a biblioteca COIN, em relação à execução da versão 6 no processador *Core2Quad.*

Não ocorreram com o processador *Nehalem* as grandes deteriorações da versão 5 ao longo do processo iterativo, verificadas principalmente nos casos com parâmetros aumentados e biblioteca COIN nos processadores *Core2Quad*. Em alguns casos ocorreram quedas nas eficiências ao longo do processo iterativo, porém, esta queda está relacionada com o processo de solução da estratégia de paralelização em si, como por exemplo, o consumo de tempo do agrupamento dos cortes, e não com algum problema no processador.

Capítulo 7.
Conclusões e Trabalhos Futuros

7.1 Conclusões

Este trabalho apresentou uma metodologia otimizada de processamento paralelo para resolver o problema de planejamento da operação de sistemas hidrotérmicos em ambientes baseados em *cluster* de computadores. O método para a solução deste problema foi a PDDE e foi utilizado como base o modelo computacional NEWAVE, homologado pela ANEEL e adotado pelo Operador Nacional do Sistema no planejamento da operação energética de médio prazo, pela Empresa de Pesquisas Energéticas nos estudos do planejamento da expansão e demais empresas do Setor Elétrico Nacional em seus estudos de planejamento energético.

A metodologia está baseada na característica de solução da PDDE de resolver diversos PLs, cada um representando um cenário de afluência, de forma independente. A distribuição dos problemas é feita de forma dinâmica, através de um processo independente que se comunica com os outros processos responsáveis pela solução dos problemas, informando qual o PL que deverá ser resolvido. Esta forma de repartir os problemas permite um melhor balanceamento de carga, proporcionando uma melhor eficiência à estratégia de paralelização.

Outras otimizações foram feitas na estrutura paralela da estratégia proposta, sempre visando melhorar o desempenho da mesma, tais como: troca de comunicação por armazenamento local no gerenciamento dos cortes de Benders; modificação da maneira de se executar o agrupamento dos cortes de Benders; e modificação de mensagens de forma a aproveitar a arquitetura de processadores com múltiplos núcleos. Além disto, o código seqüencial da aplicação, onde se implementou a estratégia de

paralelização, foi otimizado de forma a também diminuir o tempo de processamento. Em todo este processo de otimização, o procedimento para identificar, analisar e encontrar soluções que tornem possível a redução do tempo final de processamento sempre teve em mente as fases que compõem o ciclo de desempenho: obter informações de desempenho; analisar e identificar os problemas; encontrar soluções mais eficientes para realizar as mesmas tarefas; implementar estas soluções; e executar a nova versão para se verificar o ganho de eficiência e, se necessário, realizar um novo ciclo de desempenho.

A metodologia desenvolvida para criar e otimizar uma estratégia de paralização, que pode ser considerada uma das contribuições deste trabalho, consistiu nos seguintes passos:

- Otimização do código sequencial da aplicação;
- Identificação das tarefas que podem ser executadas de forma independente, de forma a distribuí-las entre os processadores participantes do ambiente de processamento paralelo (criação de uma estratégia de paralelização);
- Otimização da estratégia de paralelização, principalmente através da análise dos seguintes pontos.
    - Diminuição da comunicação entre os processos;
    - Balanceamento de carga;
    - Introdução de assincronismo;
    - Adequação à arquitetura de *hardware*.

    Caso seja necessária a utilização de bibliotecas externas, é importante descobrir quais são as mais eficientes ou as que permitam a utilização da estratégia de paralelização de forma mais eficiente. No caso específico deste trabalho, a biblioteca COIN em si não foi mais eficiente que a biblioteca OSL, porém permitiu a utilização da aplicação em ambientes de 64 bits, onde as execuções levaram significativamente menos tempo.

Um fato interessante ocorrido ao longo do trabalho foi a identificação da perda de desempenho computacional do processador Intel *Core2Quad*, principalmente quando o processo iterativo de solução levava uma grande quantidade de iterações para convergir. Este problema ocorreu tanto na execução da aplicação com 32 bits quanto

com 64 bits, porém a perda de desempenho foi mais evidente neste último caso. Estes problemas não foram identificados nas execuções com a família de processadores mais recente, chamada de *Nehalem.*

Outro fato a destacar foi o ganho de tempo computacional permitido pela estratégia de paralelização desenvolvida. A execução seqüencial do caso PMO de Março de 2009, adotado como caso base, levou 24h 55min (89.702s). Depois das várias melhorias realizadas no código fonte e com a versão inicial da estratégia de paralelização, a execução com 1 processador caiu para 24h 39min (88.733s), para 3h 37min (13.045s) com 8 processadores e o menor tempo foi 45min (2.712s) com 200 processadores. Com as várias otimizações na estratégia de paralelização, os tempos da versão paralela final caíram para 24h 27min (88.045s) com 1 processador, 3h 13min (11.591s) com 8 processadores e o menor tempo de execução foi de 23min (1.378s) com 200 processadores. Todos estes resultados, obtidos com a biblioteca OSL originalmente utilizada pelo programa NEWAVE, estão apresentados na Tabela 43.

Tabela 43 – Tempos Médios de Processamento da Estratégia de Paralelização com a Biblioteca OSL

Qte.	Processador *Core2Quad*								Processador *Nehalem*
	Seq. Inicial	Seq. Perfilada	Versão Inicial	Versão 2	Versão 3	Versão 4	Versão 5	Versão 6	Versão 5
1	24h55min	24h39min	24h32min	24h29min	24h30min	24h28min	24h27min	-	20h20min
8	-	-	3h37min	3h36min	3h27min	3h26min	3h26min	3h13min	2h47min
16	-	-	2h10min	2h00min	1h51min	1h50min	1h50min	1h43min	-
32	-	-	1h38min	1h11min	1h04min	1h02min	1h01min	58min	-
64	-	-	1h15min	46min	40min	38min	37min	35min	-
128	-	-	56min	32min	29min	27min	27min	26min	-
200	-	-	45min	27min	27min	27min	23min	-	-

Com a utilização da versão de 64 bits da biblioteca COIN, foi possível gerar um código em 64 bits do programa e com a versão paralela final os tempos caíram para 14h 52min (53.509s) com 1 processador, 2h 14min (8.020s) com 8 processadores e 15min (907s) com 200 processadores. Parte da diferença obtida com a biblioteca COIN foi devido à quantidade menor de iterações da convergência, porém, também se pode dizer que as duas soluções foram estatisticamente equivalentes. Logo, pode-se considerar que o caso PMO de Março de 2009 pôde ser resolvido no tempo de 15 min. Passando as execuções para o novo processador *Nehalem*, os tempos caíram para 11h 36min (41.775s) com 1 processador e 1h 37min (5.791s) com 8 processadores, conforme pode ser visto na Tabela 44.

Tabela 44 - Tempos de Processamentos da Estratégia de Paralelização com a Biblioteca COIN

Qte.	Processador *Core2Quad*		Processador *Nehalem*
	Versão 5	Versão 6	Versão 5
1	14h52min	-	11h36min
8	2h14min	2h01min	1h37min
16	1h11min	1h4min	-
32	39min	36min	-
64	25min	23min	-
128	18min	17min	-
200	15min	-	-

O que os tempos anteriores significam?

A primeira conclusão é que, para se obter a versão mais eficiente de uma aplicação deve-se procurar levar ao limite máximo a otimização, tanto do código fonte, quanto da estratégia de paralelização, além de procurar utilizar bibliotecas, caso existam, mais eficientes ou, pelo menos, que permitam executar a aplicação de forma mais eficiente. A simulação de um caso que originalmente levou quase 25 horas sequencialmente passou a levar, com a estratégia de paralelização, 23 minutos, redução de 65 vezes (1,54% do tempo da versão seqüencial inicial) com a biblioteca OSL, ou 15 minutos em paralelo, uma redução de 100 vezes (1,01% do tempo da versão seqüencial inicial) no tempo de execução. Essa redução só não foi maior por conta dos recursos computacionais disponíveis. Os tempos de processamento obtidos nas diversas etapas deste trabalho estão apresentados em forma relativa, onde o valor base adotado foi o tempo médio das execuções do código seqüencial original do programa, na Tabela 45, para a versão com a biblioteca OSL, e na Tabela 46, para a versão com a biblioteca COIN de 64 bits.

Tabela 45 - Tempos Médios Relativos Obtidos com a Estratégia de Paralelização e Biblioteca OSL

Qte.	Processador *Core2Quad*								Processador *Nehalem*
	Seq. Inicial	Seq. Perfilada	Versão Inicial	Versão 2	Versão 3	Versão 4	Versão 5	Versão 6	Versão 5
1	100,00	98,92	98,46	98,25	98,29	98,18	98,15	-	81,62
8	-	-	14,54	14,46	13,84	13,76	13,76	12,92	11,18
16	-	-	8,67	8,05	7,41	7,35	7,33	6,90	-
32	-	-	6,57	4,72	4,26	4,14	4,05	3,86	-
64	-	-	5,01	3,06	2,69	2,53	2,51	2,37	-
128	-	-	3,75	2,13	1,96	1,79	1,82	1,73	-
200	-	-	3,02	1,79	1,82	1,83	1,54	-	-

Tabela 46 - Tempos Médios Relativos Obtidos com a Estratégia de Paralelização e Biblioteca COIN

Qte.	Processador *Core2Quad*		Processador *Nehalem*
	Versão 5	Versão 6	Versão 5
1	59,65	-	46,57
8	8,94	8,08	6,38
16	4,73	4,29	-
32	2,63	2,42	-
64	1,67	1,52	-
128	1,22	1,13	-
200	1,01	-	-

A segunda conclusão é que a evolução da arquitetura e da velocidade de processamento das sucessivas famílias de processadores deve ser explorada, visando à redução dos tempos computacionais finais. Para se ter uma idéia do que poderia ter sido o tempo deste mesmo caso utilizando 200 processadores da família *Nehalem*, tomando como base o tempo com 8 processadores e supondo que a mesma redução de 27,8% (de 8.020s para 5.791s) em relação ao processador *Core2Quad* se mantivesse, a execução duraria por volta de 11 minutos.

É importante ressaltar também o caráter prático deste trabalho, uma vez que o mesmo poderá ter adoção pelo Setor Elétrico Brasileiro quase que imediata, disponibilizando uma aplicação significativamente mais eficiente, permitindo às instituições executar seus estudos com muito mais eficácia.

7.2 Continuação do Trabalho

Com a rápida proliferação de processadores com vários núcleos de processamento (*multicores*), a implementação de uma metodologia híbrida com a utilização de MPI, para as comunicações entre os nós, e de OpenMP, para aproveitar a característica de memória compartilhada existente entre os vários núcleos dos processadores atuais, se torna bastante interessante. A implementação feita neste trabalho de adequação ao *hardware*, utilizando dois níveis de mensagens, o primeiro entre os nós das placas *blade* do *cluster*, e o segundo entre os núcleos de processamento existente em cada nó, ajudará bastante neste futuro desenvolvimento, já que a separação das mensagens já está feita.

Outro futuro trabalho é adequar esta estratégia de paralelização para executar e avaliar o desempenho da solução dos PLs em ambientes de computação em nuvem,

assim como em placas gráficas *multicore* (GPU), aproveitando a grande capacidade de processamento existente nestas placas atualmente e a tendência de adoção das mesmas também para processamento científico.

Capítulo 8.

Referências Bibliográficas

[1] QUINN, M.J., *Parallel Programming in C with MPI and OpenMP,* McGraw-Hill Professional, 2004, ISBN-13:978-0-07-058201-9, ISBN-10:0-07-058201-7.

[2] www.top500.org, Sítio da rede mundial de computadores.

[3] www.mme.gov.br, Sítio do Ministério de Minas e Energia na rede mundial de computadores;

[4] MACEIRA, M.E.P., TERRY, L.A., COSTA, F.S., *et al.,* "Chain of Optimization Models for Setting the Energy Dispatch and Spot Price in the Brazilian System", *Power System Computation Conference*, *PSCC'02*, Sevilla, Spain, Junho de 2002.

[5] PEREIRA, M.V.F., PINTO, L.M.V.G., "Stochastic Optimization of a Multireservoir Hydroelectric System – A Decomposition Approach", *Water Resources,* Vol. 21, N°. 6, pp 779-792, 1985.

[6] PEREIRA, M.V.F., PINTO, L.M.V.G., "Multi-stage Stochastic Optimization Applied to Energy Planning", *Mathematical Programming 52*, pp 359-375, North-Holland, 1991.

[7] MACEIRA, M.E.P., DUARTE, V.S., PENNA, D.D.J., *et al*., "Ten Years of Application of Stochastic Dual Dynamic Programming in Official and Agent Studies in Brazil – Description of the Newave Program", *16th Power Systems Computation Conference (PSCC),* Glasgow, Scotland, 2008.

[8] www.memoria.eletrobras.com, Sítio da rede mundial de computadores destinado à memória da eletricidade.

[9] CABRAL, L.M.M., CACHAPUZ, P.B.B., "A Eletrobrás e a Operação dos Sistemas Elétricos Interligados Brasileiros", Centro da Memória da Eletricidade no Brasil, Coordenadoria de Pesquisa, Livro Eletrônico (*e-book*), 2000.

[10] ARARIPE NETO, T.A., KLIGERMAN, A.S., SILVA, M.N., *et al.,* "A Experiência do GCOI na Coordenação do Planejamento da operação Energética do Sistema Hidrotérmico Brasileiro", *VIII Seminário Nacional de Produção e*

Transmissão de Energia Elétrica, *VIII SNPTEE*, Operação de Sistemas Elétricos (GOP), São Paulo, SP, Brasil, Maio de 1986.

[11] TERRY, L.A., GOMES, F.B.M., ARAÚJO, L.E., *et al.,* "Modelo a Sistema Equivalente para Simulação de usinas Hidráulicas e Térmicas", *II Seminário Nacional de Produção e Transmissão de Energia Elétrica*, *II SNPTEE*, Belo Horizontes, MG, Brasil, Maio de 1973.

[12] KLIGERMAN, A.S., *Operação Ótima de Subsistemas Hidrotérmicos Interligados Utilizando Programação Dinâmica Estocástica Dual,* Tese de Msc., FEE/UNICAMP, Campinas, SP, Brasil, Fevereiro de 1992.

[13] TERRY, L.A., PEREIRA, M.V.F., ARARIPE NETO, T.A., *et al.,* "Coordinating the Energy Generation of the Brazilian National Hydrothermal Electrical Generating System", *Interfaces*, Vol. 16, No. 1, pp. 16-38, Estados Unidos, 1986.

[14] SOARES FILHO, S., "Planejamento da Operação de Sistemas Hidrotérmicos", *Revista SBA: Controle de Automação*, Vol. 1, No. 2, pp. 122-131.

[15] TYLAVSKY, D.J., BOSE, A. *et al.,* "Parallel Processing in Power Systems Computation", *IEEE Transactions on Power Systems*, Vol. 7, No. 2, pp 629-638, May 1992.

[16] FALCÃO, D.M., "Parallel and Distributed Processing Applications in Power System Simulation and Control", *Revista da Sociedade Brasileira de Automática*, Volume 5, pp. 125-143, 1994.

[17] FERLIN, E.P., *Avaliação de Métodos de Paralelização Automática,* Tese de M.Sc., IFSC/USP, São Carlos, SP, Brasil, 1997.

[18] FELTRIN, A.P., *Solução de Equações de Rede de Energia Elétrica em Computadores Multiprocessadores,* Tese de Dsc., FEE/UNICAMP, Campinas, SP, Brasil, Maio de 1991.

[19] VALE, M.H.M, FALCÃO, D.M., KASZKUREWICZ, "Electrical Power Network Decomposition for Parallel Computations", *International Symposium on Circuits and Systems, Proceedings of IEEE International Symposium on Circuits and Systems*. v. 6. p. 2761-2764, San Diego, CA, USA, 1992.

[20] LAU, K., TYLAVSKY, D.J., BOSE, A., "Coarse Grain Scheduling in Parallel Triangular Factorization and Solution of Power System Matrices", *IEEE Transactions on Power Systems*, Vol. 6, No. 2, pp. 708-714, May 1991.

[21] BORGES, C.L.T., *Avaliação da Confiabilidade Composta de Sistemas de Potência em Ambientes Computacionais Paralelos e Distribuídos,* Tese de Dsc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 1998.

[22] BORGES, C.L.T., FALCÃO, D.M., "Power System Reliability by Sequential Monte Carlo Simulation on Multicomputers Platforms", *Lecture Notes in Computer Science*, Springer-Verlag Berlim Germany, v. 1981, p. 242-253, 2001.

[23] BORGES, C.L.T., FALCÃO, D.M., "Avaliação da Confiabilidade de Sistemas de Potência em Paralelo usando Simulação Monte Carlo Sequencial", *Revista Brasileira de Controle & Automação (SBA)*, Brasil, v. 11, n. 2, p. 94-99, 2000.

[24] BORGES, C.L.T., FALCÃO, D.M., MELLO, J.C.O., *et al.*, "Composite Reliability Evaluation by Sequential Monte Carlo Simulation on Parallel and Distributed Processing Environments Parallel and Distributed State Estimation", *IEEE Transactions on Power Systems*, Vol. 16, No. 2, pp 203-209, May 2001.

[25] TEIXEIRA, M.J., PINTO, H.J.C.P., PEREIRA, M.V.F., *et al.,* "Developing Concurrent Processing Applications to Power System Planning and Operations", *IEEE Transactions on Power Systems*, Volume 5, no. 2, pp. 659-664, May 1990.

[26] BORGES, C.L.T., FALCÃO, D.M., TARANTO, G.N., "Cluster Based Power System Analysis Applications", *IEEE International Conference on Cluster Computing*, *Proceedings of IEEE International Conference on Cluster Computing*, pp. 193-200, Chemnitz, Alemanha, 2000.

[27] MOREIRA JUNIOR, C.V., *Paralelização do Modelo Computacional PLANTAC Utilizando o Ambiente OpenMP,* Dissertação de Mestrado COPPE/UFRJ, Março de 2011.

[28] ALVES, J.M.T., *Fluxo de Potência Ótimo com Restrições de Segurança Aplicado à Operação em Tempo Real Utilizando Processamento Distribuído,* Dissertação de Mestrado COPPE/UFRJ, Abril de 2005.

[29] BORGES, C.L., ALVES, J.M.T., "Power System Real Time Operation based on Security Constrained Optimal Power Flow and Distributed Processing", *IEEE Power Tech 2007*, pp 960-965, Lausanne, 1-5 July 2007.

[30] ALVES, A.C.B., *Processamento Distribuído Aplicado à Análise de Segurança Estática de Sistemas de Energia Elétrica,* Tese de Dsc., FEEC/UNICAMP, Campinas, SP, Brasil, Agosto de 1997.

[31] MÉNDEZ, O.S., *Solução Concorrente do Problema do Fluxo de Potência Ótimo com Restrições de Segurança,* Tese de Dsc., FEE/UNICAMP, Campinas, SP, Brasil, Junho de 1993.

[32] DECKER, I.C., FALCÃO, D.M., KASZKUREWICZ, E., "Parallel Implementation of a Power System Dynamic Simulation Methodology Using the Conjugate Gradient Method", *IEEE Transactions on Power Systems*, Vol. 7, No. 1, pp 458-465, February 1992.

[33] DECKER, I.C., FALCÃO, D.M., KASZKUREWICZ, E., "Conjugate Gradient Methods for Power System Dynamic Simulation on Parallel Computers", *IEEE Transactions on Power Systems*, Vol. 11, No. 3, pp 1218-1227, August 1996.

[34] LA SCALA, M., BOSE, A., TYLAVSKY, D.J., *et al.,* "A Highly Parallel Method for Transient Stability Analysis", *IEEE Transactions on Power Systems*, Vol. 5, No. 4, pp 1439-1446, November 1990.

[35] LA SCALA, M., SBRIZZAI, R., TORELLI, F., "A Pipelined-in-Time Parallel Algorithm for Transient Stability Analysis", *IEEE Transactions on Power Systems*, Vol. 6, No. 2, pp 715-722, May 1991.

[36] LEE, S.Y., CHIANG, H.D., LEE, K.G., *et al.,* "Parallel Power System Transient Stability Analysis on Hypercube Multiprocessors", *IEEE Transactions on Power Systems*, Vol. 6, No. 3, pp 1337-1343, August 1991.

[37] CHAI, J.S., ZHU, N., BOSE, A., *et al.,* "Parallel Newton Type Methods for Power System Stability Analysis Using Local and Shared Memory Multiprocessors", *IEEE Transactions on Power Systems*, Vol. 6, No. 4, pp 1539-1545, November 1991.

[38] CASTRO, M.S., *Análise On-line da Estabilidade Transitória de Sistemas Elétricos de Potência Usando Ambientes de Computação Distribuída,* Dissertação de Mestrado (M.Sc.), FEE/UNICAMP, Campinas, SP, Brasil, Outubro de 1995.

[39] GUO, Z.J., SHI, L.B., YAO, Z.X., "Distributed Parallel Computing Architecture Design Philosophy for TTC Evaluation with Transient Stability Constraints", *International Conference on Electric Utility Deregulation, Restructuring, and Power Technology*, *DRPT2008,* pp. 783-787, Nanjing, China, April 2008.

[40] JIKENG, L., XINYU, T., XUDONG, W., *et al.,* "Parallel Simulation for the Transient Stability of Power System", *International Conference on Electric Utility Deregulation, Restructuring, and Power Technology*, *DRPT2008,* pp. 1325-1329, Nanjing, China, April 2008.

[41] JALILI-MARANDI, V., DINAVAHI, V., "Instantaneous Relaxation-Based Real-Time Transient Stability Simulation", *IEEE Transactions on Power Systems*, Vol. 24. No. 3, pp. 1327-1336, August 2009.

[42] SHI, L., GUO, Z., NI, Y., *et al.,* "Implementation of a Distributed Parallel Computing Architecture for Transient Stability Constrained TTC Evaluation", *IEEE Power & Energy Society General Meeting*, *PES´09,* pp. 1325-1329, Calgary, Canada, October 2009.

[43] JALILI-MARANDI, V., DINAVAHI, V., "SIMD-Based Large-Scale Transient Stability Simulation on the Graphics Processing Unit", *IEEE Transactions on Power Systems*, Vol. 25. No. 3, pp. 1589-1599, August 2010.

[44] VIEIRA JUNIOR, J.C.M., *Método Zbus Gauss Paralelo para Cálculo de Fluxo de Potência Trifásico em Redes Assimétricas de Distribuição de Energia Elétrica,* Dissertação de Mestrado (M.Sc.), FEEC/UNICAMP, Campinas, SP, Brasil, Maio de 1999.

[45] ALVES, F.R.M., *Aplicação de Buscas Heurísticas ao Problema de Determinação de Rotas para Recomposição Fluente de Sistemas Elétricos de Potência,* Tese de Doutorado (D.Sc.), COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Abril de 2007.

[46] ROMERO, S.P., *Decomposição Lagrangeana Aplicada ao Problema de Planejamento da Expansão de Sistemas de Transmissão de Energia Elétrica Considerando Cenários de Incertezas,* Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Julho de 2007.

[47] OLIVEIRA, S.A., *Metaheurísticas Aplicadas ao Planejamento da Expansão da Transmissão de Energia Elétrica em Ambiente de Processamento Distribuído,* Tese de Dsc., FEEC/UNICAMP, Campinas, SP, Brasil, Outubro de 2004.

[48] PALIN, M.F., *Técnicas de Decomposição de Domínio em Computação Paralela para Simulação de Campos Eletromagnéticos pelo Método de Elementos Finitos,* Tese de Dsc., EP/USP, São Paulo, SP, Brasil, 2007.

[49] TREVIZAN, M.P., *Processamento Paralelo na Simulação de Campos Eletromagnéticos pelo Método das Diferenças Finitas no Domínio do Tempo,* Tese de Msc., EP/USP, São Paulo, SP, Brasil, 2007.

[50] FALCÃO, D.M., KASZKUREWICZ, E., ALMEIDA, H.L.S, "Application of Parallel Processing Techniques to the Simulation of Power System Electromagnetic Transients", *IEEE Transactions on Power Systems*, Vol. 8, No. 1, pp 90-96, February 1993.

[51] FALCÃO, D.M., WU, F.F., MURPHY, L., "Parallel and Distributed State Estimation", *IEEE Transactions on Power Systems*, Vol. 10, No. 2, pp 724-730, May 1995.

[52] CAMPAGNOLO, J.M., MARTINS, N., PEREIRA, J.L.R., *et al.,* "Fast Small-Signal Stability Assessment Using Parallel Processing", *IEEE Transactions on Power Systems*, Vol. 9, No. 2, pp 949-956, May 1994.

[53] CAMPAGNOLO, J.M., MARTINS, N., FALCÃO, D.M, "An Efficient and Robust Eigenvalue Method for Small-Signal Stability Assessment in Parallel Computers", *IEEE Transactions on Power Systems*, Vol. 10, No. 1, pp 506-511, February 1995.

[54] SATO, F., *Um Estudo Comparativo da Análise de Curto-Circuito Probabilístico em Ambientes Paralelo e Distribuído,* Tese de Dsc., FEE/UNICAMP, São Paulo, SP, Brasil, Julho de 1995.

[55] SATO, F., GARCIA, A.V., MONTICELLI, A., "Parallel Implementation of Probabilistic Short-Circuit Analysis by the Monte Carlo Approach", *IEEE Transactions on Power Systems*, Vol. 9, No. 2, pp 826-832, May 1994.

[56] TÃO, W.K., *Minimização de Perdas em Redes de Distribuição de Energia Elétrica Através de Métodos de Busca Inteligentes com Processamento Paralelo,* Tese de Msc., FEEC/UNICAMP, Campinas, SP, Brasil, Março de 1999.

[57] BARÁN, B., KASZKUREWICZ, E., FALCÃO, D.M., "Team Algorithms in Distributed Load Flow Computations", *IEE Proceedings – Generation, Transmission and Distribution*, Vol. 142, No. 6, pp 583-588, November 1995.

[58] ALMEIDA, C.R.T., *Time Assíncrono Inicializador para o Planejamento da Expansão da Transmissão,* Tese de Msc., FEE/UNICAMP, Campinas, SP, Brasil, Março de 1998.

[59] MARIÑOS, Z.A., PEREIRA, J.L.R., CARNEIRO JR., S., "Fast harmonic power flow calculation using parallel processing", *IEE Proceedings – Generation, Transmission and Distribution*, Vol. 141, No. 1, pp. 27-32, January 1994.

[60] AVELEDA, A.A., *Utilização de Sistemas de Alto Desempenho no Processamento de Sinais na Análise de Problemas de Vibrações Induzidas por Desprendimento de Vórtices em Estruturas Offshore,* Tese de Doutorado COPPE/UFRJ, Setembro de 2003.

[61] WU-ZHI, Z., XIN-LI, S., YONG, T., *et al.,* "A Frequency-Domain Parallel Eigenvalue Search Algorithm of Power Systems Based on Multi-Processing", *Power Systems Conference & Exposition, IEEE Power & Energy Society*, Phoenix, Arizona, USA, March 20-23, 2011.

[62] MCGINN, S.F., SHAW, R.E., "Parallel Gaussian Elimination Using OpenMP and MPI", *Proceedings of the 16th Annual International Symposium on High Performance Computing Systems and Applications*, *HPCS´02*, Moncton, Canada, August 2002.

[63] SHI, A., SHEN, W., LI, Y., *et al.,* "Implementation and Analysis of Jacobi Iteration Based on Hybrid Programming", *2010 International Conference On Computer Design And Appliations*, *ICCDA 2010*, volume 2, págs 311-314, Paris, France, August 2010.

[64] FARIA JUNIOR, H., *Uma Nova Metaheurística para Problemas Combinatórios Aplicada ao Planejamento da Expansão de Sistemas de Transmissão de Energia Elétrica,* Tese de Doutorado COPPE/UFRJ, Setembro de 2005.

[65] MARCATO, A.L.M., JR, I.C.S., GARCIA, P.A.N., *et al.,* "Genetic Algorithm Approach Applied to Long Term Generation Expansion Planning", *Transmission & Distribution Conference Exposition: Latin America,* 15-18 August 2006

[66] MATAR, M., IRVANI, R., "Massively Parallel Implementation of AC Machine Models for FPGA-Based Real-Time Simulation of Electromagnetic Transients", *IEEE Transactions on Power Delivery*, Vol. 26, No. 2, pp 830-840, April 2011.

[67] LI, Y., LI, F., LI, W., "Parallel Power Flow Calculation Based on Multi-port Inversed Matrix Method", *International Conference on Power System Technology*, *POWERCON*, 24-28, October 2010.

[68] LI, Y., SHEN, W., SHI, A., *et al.,* "MPI and OpenMP Paradigms on Cluster with multicores and its application on FFT", 2010 *International Conference On Computer Design And Appliations*, *ICCDA 2010*, volume 1, págs 23-26, Paris, France, August 2010.

[69] JIA-AN, Z., NA, Z., YI-LANG, J., "Parallelizing Methods Analysis for Solving Large Sparse Power Network Equations on Multi-Core Processor Platforms", *4th International Conference on Intelligent Computation Technology and Automation*, *ICICTA*, Vol. 1, pp 192-195, March 2011.

[70] SPEYER, G., FREED, N., AKIS, R., *et al.,* "Paradigms for Parallel Computation", *DoD HPCMP Users Group Conference*, *DOD HPCMP UGC*, Vol. 1, pp 486-494, July 2008.

[71] RABENSEIFNER, R., HAGER, G., JOST, G., "Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes", *Parallel, Distributed and Network-based Processing, 17th Euromicro International Conference*, pp. 427-436, February 2009.

[72] XU, C., BAI, Y., LUO, C., "Performance Evaluation of Parallel Programming in Virtual Machine Environment", *6th International Conference on Network and Parallel Computing*, pp. 140-147, October 2009.

[73] TSUJI, M., SATO, M., "Performance evaluation of OpenMP and MPI hybrid programs on a large scale multi-core multi-socket cluster, T2K Open Supercomputer", *International Conference on Parallel Processing Workshop*, *ICPPW*, pp. 206-213, September 2009.

[74] NOAJE, G., KRAJECKI, M., JAILLET, C., "MultiGPU computing using MPI or OpenMP", *International Conference on Intelligent Computer Communication and Processing*, *ICCP*, pp 347-354, August 2010.

[75] YANG, C., HUANG, C., LIN, C., *et al.,* "Hybrid Parallel Programming on GPU Clusters", *International Symposium on Parallel and Distributed Processing with Applications*, *ISPA*, pp 142-147, September 2010.

[76] MO, B., HEGGE, J., WANGENSTEEN, I., "Stochastic Generation Expansion Planning by Means of Stochastic Dynamic Programming", *IEEE Transactions on Power Systems*, Vol. 6, No. 2, pp. 662-668, May 1991.

[77] DAPKUS, W.D., BOWE, T.R., "Planning for New Electric Generation Technologies a Stochastic Dynamic Programming Approach", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-103, No. 6, June 1984.

[78] QUINTANA, V.H., CHIKHANI, A.Y., "A Stochastic Model for Mid-Term Operation Planning of Hydro-Thermal Systems with Random Reservoir Inflows", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-100, No. 3, March 1981.

[79] SHERKAT, V.R., CAMPO, R., MOSLEHI, K., LO, E.O., "Stochastic Long-Term Hydrothermal Optimization for a Multireservoir System", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-104, No. 8, pp. 2040-2050, August 1985.

[80] LI, C., YAN, R., ZHOU, J., "Stochastic Optimization of Interconnected Multireservoir Power Systems", *IEEE Transactions on Power Systems*, Vol. 5, No. 4, pp. 1487-1496, November 1990.

[81] BOND, P.S., *Otimização Determinística Individualizada da Operação Energética do Sistema Hidro-Térmico Interligado Sul-Sudeste do Brasil, em Horizontes de Médio e Longo Prazos,* Tese de Msc., FEE/UNICAMP, Campinas, SP, Brasil, 1988.

[82] CARNEIRO, A.A.F.M., BOND, P.S., "A Large Scale Application of an Optimal Deterministic Hydrothermal Scheduling Algorithm", *IEEE Transactions on Power Systems*, Vol. 5, No. 1, pp. 204-211, February 1990.

[83] MARQUES, T.C., CIGONA, M.A., SOARES, S., "Benefits of Coordination in the Operation of Hydroelectric Power Systems: Brazilian Case", *IEEE PES General Meeting, Proceedings of the IEEE PES General Meeting*, Montreal, Canada, 2006.

[84] CRUZ JUNIOR, G., *Modelo Equivalente Não Linear para o Planejamento da Operação a Longo Prazo de Sistemas de Energia Elétrica,* Tese de Dsc., FEEC/UNICAMP, Campinas, SP, Brasil, Dezembro de 1998.

[85] ZAMBELLI , M.S., *Planejamento da Operação Energética Via Curvas-Guia de Armazenamento,* Tese de Msc., FEEC/UNICAMP, Campinas, SP, Brasil, Julho de 2006.

[86] CIGONA , M.A., *Modelo de Planejamento da Operação Energética de Sistemas Hidrotérmicos a Usinas Individualizadas Orientado a Objetos,* Tese de Msc., FEEC/UNICAMP, Campinas, SP, Brasil, Fevereiro de 1999.

[87] MARTINEZ , L., *Políticas de Controle Malha Fechada e Malha Aberta no Planejamento da Operação Energética de Sistemas Hidrotérmicos,* Tese de Dsc., FEEC/UNICAMP, Campinas, SP, Brasil, Setembro de 2001.

[88] MARTINEZ, L., SOARES, S., "Comparison Between Closed-Loop and Partial Open-Loop Feedback Control Policies in Long Term Hydrothermal Scheduling", *IEEE Transactions on Power Systems*, Vol. 17, No. 2, pp. 330-336, May 2003.

[89] MARTINEZ, L., SOARES, S., "Primal and Dual Stochastic Dynamic Programming in Long Term Hydrothermal Scheduling", *Power Systems Conference and Exposition*, New York, 2004.

[90] SIQUEIRA, T.G., *Comparação entre Programação Dinâmica Estocástica Primal e Dual no Planejamento da Operação Energética,* Tese de Msc., FEEC/UNICAMP, Campinas, SP, Brasil, Junho de 2003.

[91] MATOS, V.L., *Análise Comparativa entre as Modelagens de Reservatório Equivalente de Energia Agregado por Subsistema e por Cascata no Problema do Planejamento Anual da Operação Energética,* Tese de Msc., EE/UFSC, Florianópolis, SC, Brasil, Junho de 2008.

[92] PEREIRA, M.V.F., PINTO, L.M.V.G., "Application of Decomposition Techniques to the Mid - and Short - Term Scheduling of Hydrothermal Systems", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-102, No. 11, pp. 3611-3618, November 1983.

[93] SOUZA, B.B., *Avaliação do Impacto da Representação Explicita de Bacias Hidrográficas Através do Acoplamento Hidráulico no Planejamento da Operação Energética de Médio Prazo,* Tese de Msc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2008.

[94] LOPES, J.E.G., *Otimização de Sistemas Hidroenergéticos,* Tese de Msc., EP/USP, São Paulo, SP, Brasil, Maio de 2001.

[95] MARQUES, T.C., *Uma Política Operativa a Usinas Individualizadas para o Planejamento da Operação Energética do Sistema Interligado Nacional,* Tese de Dsc., FEEC/UNICAMP, Campinas, SP, Brasil, Dezembro de 2006.

[96] AMENDOLA, A.F., *Metaheurísticas de Otimização Aplicadas à Coordenação Hidrotérmica,* Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, Junho de 2007.

[97] HUMPIRI , C.J.P., *Estratégias Evolutivas no Planejamento Energético da Operação de Sistemas Hidrotérmicos de Potência,* Tese de Msc., FEEC/UNICAMP, Campinas, SP, Brasil, Julho de 2005.

[98] LEITE, P.T., CARNEIRO, A.A.F.M., CARVALHO, A.C.P.L.F., "Aplicação de Algoritmos Genéticos na Determinação da Operação Ótima de Sistemas Hidrotérmicos de Potência", *Revista Controle & Automação*, Vol.17 no.1, Março de 2006.

[99] ZOUMAS, C.E., BAKIRTZIS, A.G., THEOCHARIS, J.B., *et al.*, "Genetic Algorithm Solution Approach to the Hydrothermal Coordination Problem", *IEEE Transactions on Power Systems*, Vol. 19, No. 2, pp. 1356-1364, May 2004.

[100] SIFUENTES, W.S., VARGAS, A., "Hydrothermal Scheduling Using Benders Decomposition: Accelerating Techniques", *IEEE Transactions on Power Systems*, Vol. 22, No. 3, pp. 1351-1359, August 2007.

[101] CHANG, H.S., FU, M.C., MARCUS, S.I., "An Asymptotically Efficient Algorithm for Finite Horizon Stochastic Dynamic Programming Problems", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Hawaii, USA, December 2003.

[102] UNSIHUAY, C., MARANGON-LIMA, J.W., SOUZA, A.C.Z., "Short-Term Operations Planning of Integrated Hydrothermal and Natural Gas Systems", *IEEE PES Power Tech Conference*, pp. 1410-1416, 1-5, July 2007, Lausanne, Switzerland.

[103] FINARDI, E.C., *Planejamento da Operação de Sistemas Hidrotérmicos Utilizando Computação de Alto Desempenho,* Dissertação de Mestrado, UFSC, Florianópolis, SC, Brasil, Abril de 1999.

[104] SILVA, E.L., FINARDI, E.C., "Planning of Hydrothermal Systems Using a Power Plant Individualistic Representation", *IEEE Power Tech Conference, Proceedings of the IEEE Power Tech Conference*, Porto, Portugal, September 2001.

[105] SILVA, E.L., FINARDI, E.C., "Parallel Processing Applied to the Planning of Hydrothermal Systems", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 8, pp. 721-729, August 2003.

[106] BORGES, C.L.T., PINTO, R.J., "Small Hydro Power Plants Energy Availability Modeling for Generation Reliability Evaluation", *IEEE Transactions on Power Systems*, v. 23, pp. 1125-1135, 2008.

[107] BORGES, C.L.T., PINTO, R.J., "Small Hydro Power Plants Energy Availability Modeling for Generation Reliability", *IEEE Power Engineering Society General Meeting*, Calgary, Canada, 2009.

[108] PINTO, R.J., DUARTE, V.S., MACEIRA, M.E.P., *et al.*, "Metodologia para Aplicação de Processamento Distribuído no Planejamento da Expansão e Operação", *XXXIX Simpósio Brasileiro de Pesquisa Operacional, SBPO*, pp. 548-558, Fortaleza, CE, Brasil, Agosto de 2007.

[109] PINTO, R.J., BORGES, C.L.T., MACEIRA, M.E.P., "Aplicação de Processamento Distribuído no Planejamento da Expansão e Operação de Sistemas Hidrotérmicos", *XXX Iberian-Latin-American Congress on*

Computational Methods in Engineering (CILAMCE), Búzios, RJ, Brasil, Novembro de 2009.

[110] PINTO, R.J., SABÓIA, A.L.G.P., CABRAL, R.N. *et al.,* "Metodologia para Aplicação de Processamento Distribuído no Planejamento de Curto-Prazo da Operação Hidrotérmica", *XI Simpósio de Especialistas em Planejamento da Operação e Expansão Elétrica, SEPOPE*, Belém, PA, Brasil, Março de 2009.

[111] PINTO, R.J., SABÓIA, A.L.G.P., CABRAL, R.N., *et al.,* "Metodologia para Aplicação de Processamento Paralelo no Planejamento de Curto-Prazo (Modelo DECOMP)", *XX Seminário Nacional de Produção e Transmissão de Energia Elétrica, SNPTEE*, GOP-25, Recife, PE, Brasil, Novembro de 2009.

[112] KOTHARI, D.P., DHILLON, J.S., *Power System Optimization,* Prentice-Hall of India Private Limited, 2004, ISBN-10: 81-203-2197-9.

[113] SILVA, E.L., *Formação de Preços em Mercados de Energia Elétrica,* Editora Sagra Luzzatto, 1ª Edição, 2001, ISBN-13: 978-8-52410646-0, ISBN-10: 852410646-8.

[114] WOOD, A.J., WOLLENBERG, B.F., *Power Generation, Operation and Control,* John Wiley & Sons, 2ª Edição, 1996, ISBN-13:978-81-265-0838-9.

[115] ARVANITIDIS, N.V., ROSING, J., "Composite Representation of a MultireservoirHydroelectric Power SystemPower System Optimization", *IEEE Transaction on Power Apparatus and System*, vol. PAS-89, no. 2, pp. 319-326, February 1970.

[116] OLIVEIRA, G.G., *Otimização da Operação Energética de Sistemas Hidrotérmicos com Representação Individualizadas das Usinas e Afluências Determinísticas,* Tese de Msc., FEE/UNICAMP, Campinas, SP, Brasil, Setembro de 1993.

[117] MARCATO, A.L.M., *Representação Híbrida de Sistemas Equivalentes e Individualizados para o Planejamento da Operação de Médio Prazo de Sistemas de Potência de Grande Porte,* Tese de Dsc., PUC-RJ, Rio de Janeiro, RJ, Brasil, Maio de 2002.

[118] DUARTE, V.S., *Modelagem da Vazão Mínima Obrigatória em Problemas de Planejamento da Operação de Longo Prazo de Sistemas Hidrotérmicos Interligados,* Tese de Msc., UFJF, Juiz de Fora, MG, Brasil, Fevereiro de 2002.

[119] BELLMAN, R.E., *Dynamic Programming,* Courier Dover Publications, reimpressão, 2003, ISBN-10:0-486-42809-5.

[120] MARZANO, L.G.B., *Otimização de Portfólio de Contratos de Energia em Sistemas Hidrotérmicos com Despacho Centralizado,* Tese de Dsc., PUC-RJ, Rio de Janeiro, RJ, Brasil, Junho de 2004.

[121] WINKINSON, B., ALLEN, M., *Parallel Programming – Techniques and Applications Using Networked Workstations and Parallel Computers,* Prentice-Hall, 2nd Edition, 1999, ISBN-10:81-317-0239-1.

[122] PACHECO, PETER S., *Parallel Programming with MPI,* Morgan Kaufmann, 1997, ISBN-13: 978-1-55860-339-4, ISBN-10: 1-55860-339-5.

[123] DONGARRA, J., FOSTER, I., FOX, G., *et al.*, *The Sourcebook of Parallel Computing,* Morgan Kaufmann, 2003, ISBN-10: 81-8147-613-1.

[124] MATTSON, T.G., SANDERS, B.A., MASSINGILL, B., *Patterns for Parallel Programming,* Addison Wesley, 2005, ISBN-10: 0-321-22811-1.

[125] BARNEY, B., "Introduction to Parallel Computing", Trabalho disponibilizado no sítio https://computing.llnl.gov/tutorials/parallel_comp/" do Lawrence Livermore National Laboratory.

[126] FLYNN, M.J., "Very High Speed Computing System", *Proceedings of the IEEE*, Volume 54, no. 12, pp. 1901-1909, December, 1966.

[127] PFISTER, G.F., *In Search of Clusters,* Prentice-Hall, 2nd Edition, 1998, ISBN-10: 0-13-899709-8.

[128] SNIR, M., OTTO, S., HUSS-LEDERMAN, S., WALKER, *et al., MPI: The Complete Reference*, MIT Press, 1996.

[129] GROPP, W., EWING, L., SKJELLUM, A., *Using MPI: Portable Parallel programming with the Message Passing Interface,* 2nd Edition, MIT Press, 1999, ISBN-13: 978-0-262-57134-0, ISBN-10: 0-262-57134-X.

[130] MPI Forum, *MPI-2: Extensions to the Message-Passing Interface,* Documento disponibilizado de forma eletrônica através do arquivo "mpi-20.pdf", 1997.

[131] GROPP, W., EWING, L., THAKUR, R., *Using MPI-2: Advanced Features of the Message Passing Interface,* MIT Press, 1999, ISBN-13: 978-0-262-57133-3, ISBN-10: 0-262-57133-1.

[132] www.openmp.org, Sítio da rede mundial de computadores, http://www.openmp.org/mp-documents/paper/node4.html.

[133] CHANDRA, R., DAGUN, L., KOHR, D., MAYDAN, D., MCDONALD, J., MENON, R., *Parallel Programming in OpenMP,* Morgan Kaufmann Publishers, 2001, ISBN-10: 1-55860-671-8.

[134] CHAPMAN, B., JOST, G., VAN DER PAS, R., *Using OpenMP: Portable Shared Memory Parallel Programming,* MIT Press, 2008, ISBN-13: 978-0-262-53302-7.

[135] SANDERS, J., KANDROT, E., *CUDA by Example: An Introduction to General-Purpose GPU Programming*, 1st Edition, Addison-Wesley Pearson Education, 2010.

[136] MILLER, R., BOXER, L., *Algorithms Sequential & Parallel – A Unified Approach*, 2nd Edition, Charles River Media Inc., 2005, ISBN-10: 1-58450-412-9.

[137] RAUBER T., RUNGER, G., *Parallel Programming for Multicore and Cluster Systems*, Springer-Verlag, 2010, ISBN-13: 978-3-642-04817-3.

[138] PACHECO, P., *An Introduction to Parallel Programming*, Morgan Kaufmann Publishers, 2011 ISBN-13: 978-0-12-374260-5.

[139] HAGER, G., WELLEIN G., *Introduction to High Performance Computing for Scientists and Engineers,* CRC Press, 2011, ISBN-13: 978-1-4398-1192-4.

[140] AMDAHL, G. M., "Validity of the single-processor approach to achieving large scale computing capabilities", *AFIPS Conference Proceedings vol. 30*, (Atlantic City, N.J., Apr. 18-20), AFIPS Press, Reston, Va., 1967, pp.483-485.

[141] GUSTAFSON, J.L., "Reevaluating Amdahl´s Law", *Communications of the ACM, Nota Técnica*, Volume 31, Número 5, pp. 532-533, Maio de 1998.

[142] MORTON, D.P., "An Enhanced Decomposition Algorithm for Multistage Stochastic Hydroelectric Scheduling", *Annals of Operations Research 64*, pp. 211-235, Science Publishers, 1996.

[143] ELIAS, R.N., CAMATA, J.J, AVELEDA, A.A. *et al*., "Evaluation of Message Passing Comunication Patterns in Finite Element Solution of Coupled Problems", *9th International Meeting High Performance Computing for Computacional Science*, *VECPAR 2010*, Berkeley, California, USA, 22-25 June 2010.

Anexo A

Detalhes da Implementação da Metodologia Paralela no Modelo Newave

A.1 Estudo do Código Sequencial

O problema de planejamento da operação de médio prazo é resolvido no modelo Newave por programação dinâmica dual estocástica (PDDE). Neste programa, os cálculos necessários para a execução dos ciclos *backward* e *forward* são realizados em rotinas que recebem o mesmo nome do respectivo ciclo.

Após a convergência do processo iterativo, composta de várias passagens pelos ciclos *backward* e *forward*, o programa Newave realiza uma simulação final, que consiste num ciclo *forward*, executado pela mesma rotina *forward* utilizada no processo iterativo, só que com uma quantidade muito maior de cenários de afluências. O objetivo da simulação final é obter estatísticas mais refinadas do custo de operação, valor da água, riscos de não atendimento à demanda, além de outras variáveis ligadas à solução do problema de planejamento da operação.

Como será mostrado a seguir, o estudo do código do programa Newave visa a determinação das estruturas das duas rotinas principais do cálculo da solução do problema de planejamento da operação (rotinas *backward* e *forward*), de forma a identificar os pontos onde é possível a aplicação de técnicas de processamento paralelo para a distribuição de tarefas concorrentes.

A estrutura da rotina *backward*, mostrada no fluxograma da Figura 101, consiste em três grandes laços: o primeiro corresponde ao período pós-estudo, que tem o objetivo de anular a influência do custo futuro nulo do último período; o segundo corresponde ao período de estudo propriamente dito; o terceiro consiste no período pré-

estudo, que tem o objetivo de eliminar a influência do armazenamento inicial. Como estes três laços são similares, somente um deles foi representado no fluxograma. Outra característica importante desta rotina é que, para melhorar a representatividade dos cortes de Benders, em vez de utilizar apenas um cenário de energia afluente por série hidrológica, são utilizados vários cenários e o corte final é um corte médio gerado por esta abertura de cenários.

A estrutura da rotina *forward* é similar à da rotina *backward*, conforme está mostrado na Figura 102. Esta rotina também possui três grandes laços, representando os períodos pré-estudo, estudo e pós-estudo, com a diferença que a ordem de execução destes períodos é invertida em relação à ordem de execução na rotina *backward*. Esta rotina é utilizada em duas situações: na convergência do processo iterativo para obter o valor esperado do custo de operação; na simulação final, onde são calculados os índices de desempenho, tais como a média do custo de operação, os custos marginais, os riscos de *deficit* etc. Nestes cálculos não é necessária a abertura de cenários utilizada na rotina *backward*, resultando numa estrutura mais simples.

Convém ressaltar que os problemas que são resolvidos dentro de um mesmo período, referentes aos cenários de afluência, podem ser resolvidos simultaneamente. Logo, a paralelização destes problemas reduz significativamente o tempo total de execução de um estudo.

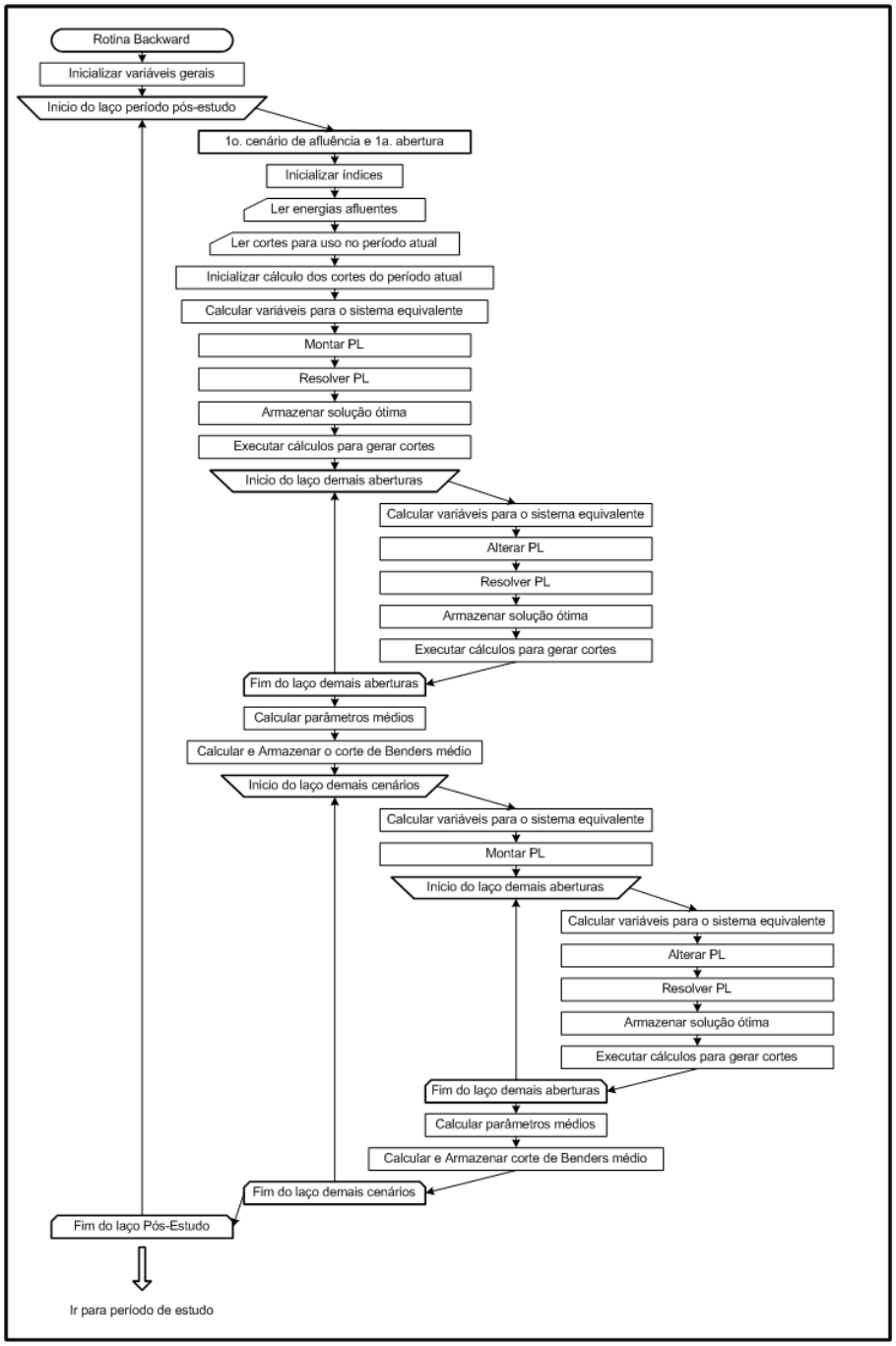

Figura 101 - Fluxograma da Rotina *Backward* (Processamento Seqüencial)

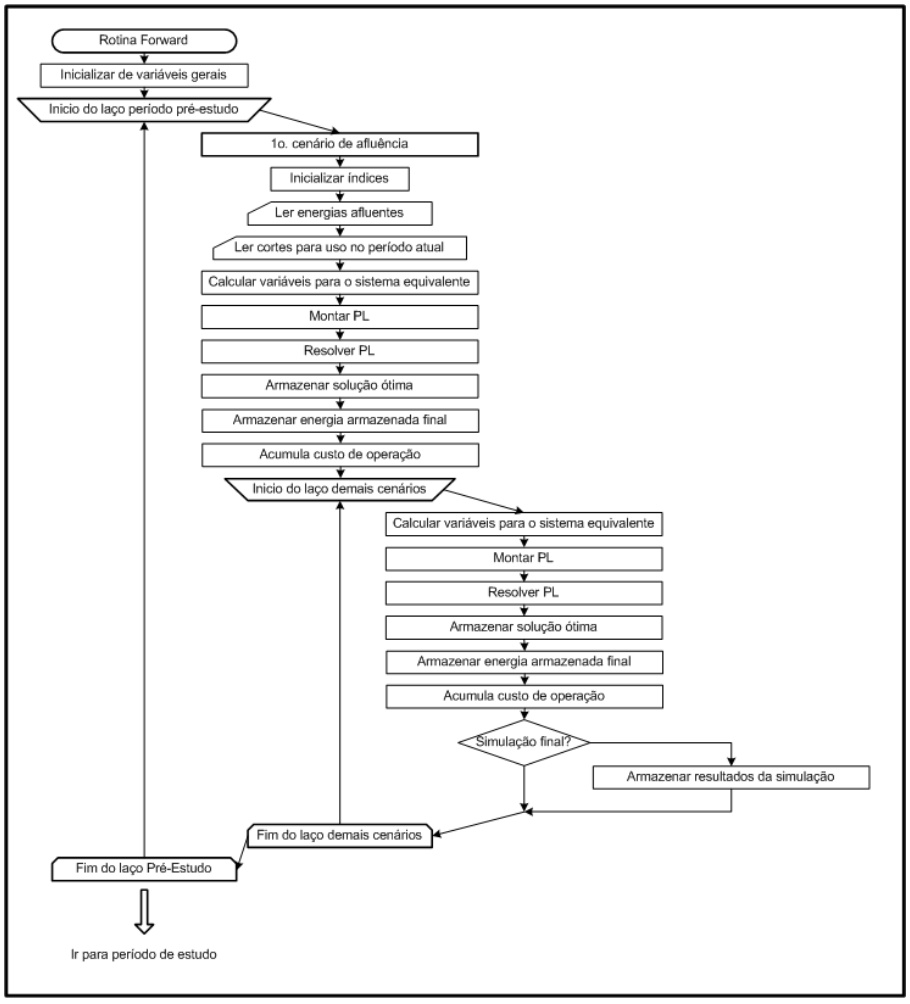

Figura 102 - Fluxograma da Rotina *Forward* (Processamento Seqüencial)

Pode-se perceber que as duas rotinas possuem processos muito similares, que consistem nas seguintes etapas:

- leitura das afluências e dos cortes a serem utilizados no período corrente. Estas leituras são pontos de sincronização, uma vez que somente o processador 0 irá ler estes dados, os mesmos deverão ser transmitidos para todos os outros processadores, que deverão estar parados a espera da chegada das mensagens;
- solução do primeiro problema. No caso da rotina *backward*, o primeiro problema se refere à primeira abertura do primeiro cenário, enquanto que na rotina *forward* ele se refere ao problema do primeiro cenário;

- solução dos problemas das demais aberturas do primeiro cenário. Esta tarefa existe apenas na rotina *backward* e consiste de um laço para resolver os outros PLs do primeiro cenário;
- solução dos demais cenários. Esta etapa consiste na solução dos PLs referentes aos outros cenários de algum dos três períodos em que se dividem as duas rotinas. Convém ressaltar que na rotina *backward*, existe mais um laço com aberturas para cada cenário de afluência.
- No caso da rotina forward, existe uma etapa de armazenamento em disco dos resultados do programa na simulação final.

No item a seguir será detalhada a estratégia de paralelização adotada neste trabalho.

A.2 Desenvolvimento da Versão Paralela

O estudo do código fonte do programa Newave visou identificar todos pontos do programa em que fosse possível distribuir o processamento para outros processadores. Basicamente, o código do programa pode ser dividido em cinco fases: (1) inicialização de arquivos e unidades lógicas; (2) leitura de dados; (3) cálculos iniciais; (4) cálculo da política de operação de sistemas hidrotérmicos, que engloba a convergência da política (processos *backward* e *forward*) e a simulação final (somente processo *forward*); (5) impressão de resultados.

Nesta versão do trabalho optou-se por realizar toda a inicialização, leitura de dados, cálculos iniciais e impressão de resultados no processador 0 (ou mestre). Outra razão para desta decisão, foi o fato de que estas etapas não gastarem muito tempo de processamento em relação ao tempo total de um caso. Logo, as alterações de código concentraram-se na etapa de convergência do cálculo da política de operação de sistemas hidrotérmicos e na simulação final.

Para executar as comunicações entre os processadores, adotou-se a biblioteca de funções MPICH2, que é uma implementação do padrão MPI, disponibilizada no sítio "http://www.mcs.anl.gov/research/projects/mpich2" da rede mundial de computadores.

A Figura 103 apresenta, de forma simplificada, um fluxograma do programa Newave, destacando os pontos de início/fim do MPI, transmissão dos dados/resultados e onde houve a distribuição das tarefas do processamento distribuído.

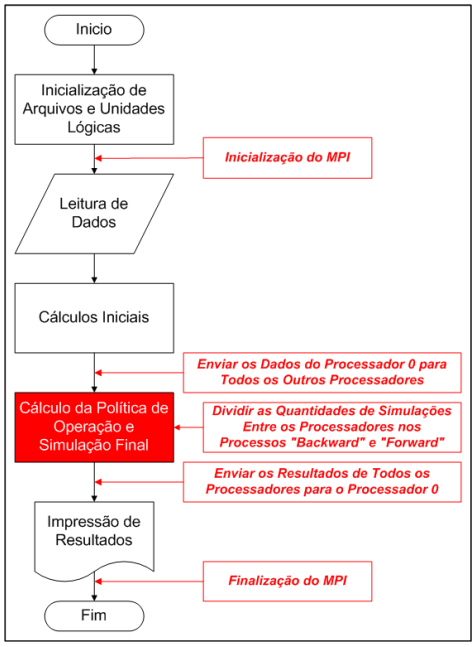

Figura 103 – Fluxograma Simplificado do Programa NEWAVE

Para alterar o código fonte foi necessário entender como que seria o funcionamento do programa na forma distribuída. O mesmo arquivo executável será utilizado em todos os processadores participantes do MPI. Ou seja, todos os processadores executam as mesmas instruções, a menos que sejam identificados os trechos do código privativos de um ou de um grupo de processadores. Nestes trechos foram utilizadas instruções de teste condicional ("IF"), permitindo que apenas os processadores desejados executassem o conjunto de instruções. Este modelo de programação é o SPMD (*Single Program Multiple Data*).

O primeiro trabalho a ser feito foi a identificação das variáveis necessárias para que os processadores, que não fossem o mestre, pudessem montar os PLs de forma correta. Isto foi necessário porque somente o processador mestre tem acesso às variáveis disponíveis nos arquivos de dados. As instruções para envio destas variáveis foram implementadas no programa principal, antes do início da rotina que gerencia o processo iterativo *backward-forward*.

A etapa seguinte foi analisar a convergência da política de operação, que consiste na execução de sucessivos processos *backward* (estimar o valor esperado do custo de operação) e *forward* (simular a operação do sistema), até que se encontre a convergência do processo. Os cálculos destes processos são claramente distribuíveis,

uma vez que são independentes entre si, com exceção dos cortes de Benders que precisam ser os mesmos em qualquer processador. Logo, este fato cria um acoplamento entre os processos que estão sendo executados. No ciclo *backward* os valores necessários no cálculo destes cortes são transferidos, ao final de cada período, para o processador mestre, onde são processados e, no inicio do período seguinte, os cortes calculados são enviados de volta para todos os processadores. No ciclo *forward* o processo consiste somente no envio a cada período dos cortes de Benders calculados para todos os processadores participantes do ambiente de processamento paralelo.

A.3 Implementação Utilizando Instruções MPI

O modelo concebido para a distribuição de tarefas adotou que toda a parte serial do código, que corresponde à leitura de dados, cálculos iniciais e impressão dos resultados, seria sempre executada pelo processador 0 (ou mestre) e este mesmo processador participaria em igualdade de condições com todos os outros nas tarefas que fossem distribuídas. Este modelo pode ser visualizado na Figura 104, Figura 105 e na Figura 106.

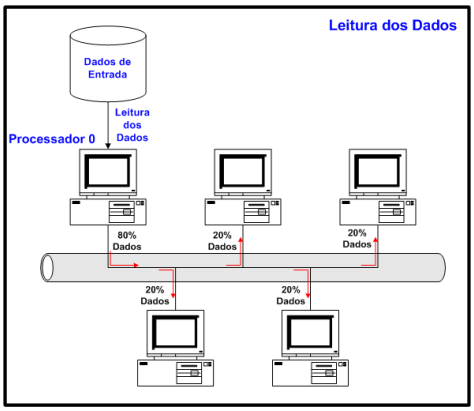

Figura 104 – Esquema de Leitura e Envio dos Dados

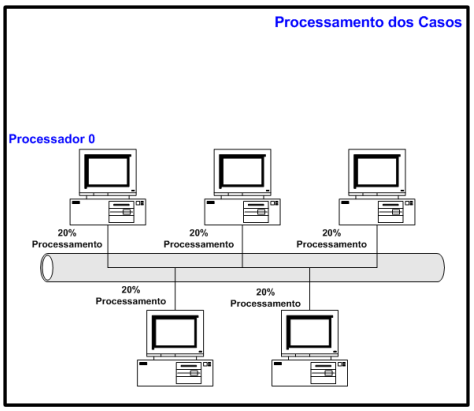

Figura 105 – Esquema de Processamento do Caso

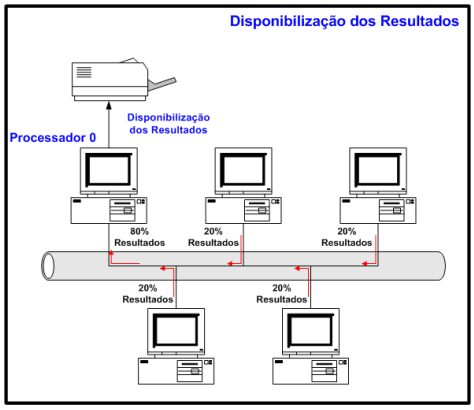

Figura 106 – Esquema de Recebimento e Impressão dos Resultados

- **Inicialização**

A inicialização do ambiente do MPI foi feita através das seguintes instruções: *MPI_INIT*, para inicializar o ambiente MPI; *MPI_COMM_RANK*, para obter a identificação do processador; e *MPI_COMM_SIZE*, para obter a quantidade de processadores participantes do processamento distribuído.

- **Envio dos Dados do Processador 0 para Todos os Outros Processadores**

Como a leitura dos dados é feita por apenas um processador, todas as variáveis necessárias para a execução do cálculo da política de operação e da simulação final devem ser enviadas para todos os processadores participantes do processamento paralelo. A função do MPI que foi utilizada para transmitir os dados foi a *MPI_BCAST*.

- **Dividir as Quantidades de Simulações Entre os Processadores nos Processos *Backward* e *Forward* e na Simulação Final**

A seguir estão as descrições dos procedimentos adotados para implementar o processamento distribuído nos processos *backward* e *forward*.

- **Processo *Backward***

Este processo começa com a solução dos problemas referentes ao último período do estudo. Nesta etapa, o programa prepara e monta as matrizes do primeiro PL. Depois desta solução, o programa resolve os PLs para os outros cenários de afluência da primeira simulação do primeiro período. Por fim, o programa acumula alguns resultados

e calcula os cortes de Benders para o estágio anterior desta simulação. Estas instruções são repetidas para todos os demais períodos do horizonte de planejamento.

De forma simplificada, podem-se relacionar as etapas da seguinte maneira: (1) leitura de energias afluentes de períodos passados e do período atual; (2) leitura dos cortes de Benders para o período atual; (3) cálculos gerais; (4) inicialização do PL; (5) montagem das matrizes e solução do PL; (6) cálculo e armazenamento de resultados. Estas etapas são repetidas para as outras aberturas dos outros cenários dos outros períodos.

As adequações necessárias para permitir a execução distribuída dos cálculos desta rotina são as seguintes:

1. Concentrar todas as leituras no processador mestre. Este é o responsável pela transmissão dos dados lidos para todos os processadores. Para a transmissão dos dados das energias afluentes foram utilizadas funções *MPI_SCATTERV*, pois somente os dados referentes às simulações que serão calculadas pelo processador são significativos. Para a transmissão dos valores dos cortes de Benders do período corrente foram utilizadas funções *MPI_BCAST*;
2. Inicializar o valor da primeira simulação que será calculada pelo processador;
3. Ao final de cada simulação, em vez de agrupar os cortes de Benders calculados e adicioná-los à FCF do próximo período gradualmente, o programa passa a armazenar os valores encontrados. Esta mudança foi necessária porque somente ao final de todas as simulações é que os cortes calculados estarão disponíveis para serem transmitidos para o processador mestre;
4. Ao final de todas as simulações do período, transmitir os cortes de Benders calculados para o processador mestre. Este processador é o responsável pelo agrupamento de todos os cortes de Benders e pela incorporação destes à FCF do próximo estágio. Nas transmissões dos cortes de Benders foram utilizadas funções *MPI_GATHERV*.

Na Figura 107 e na Figura 108 estão apresentados os fluxogramas da rotina *backward* com as instruções que são executadas pelo processador mestre e pelos demais processadores, respectivamente. Estes fluxogramas podem ser comparados com a versão seqüencial mostrada na Figura 101. As diferenças entre cada um dos

fluxogramas da versão paralela e da versão seqüencial estão mostradas através dos passos grifados em cinza.

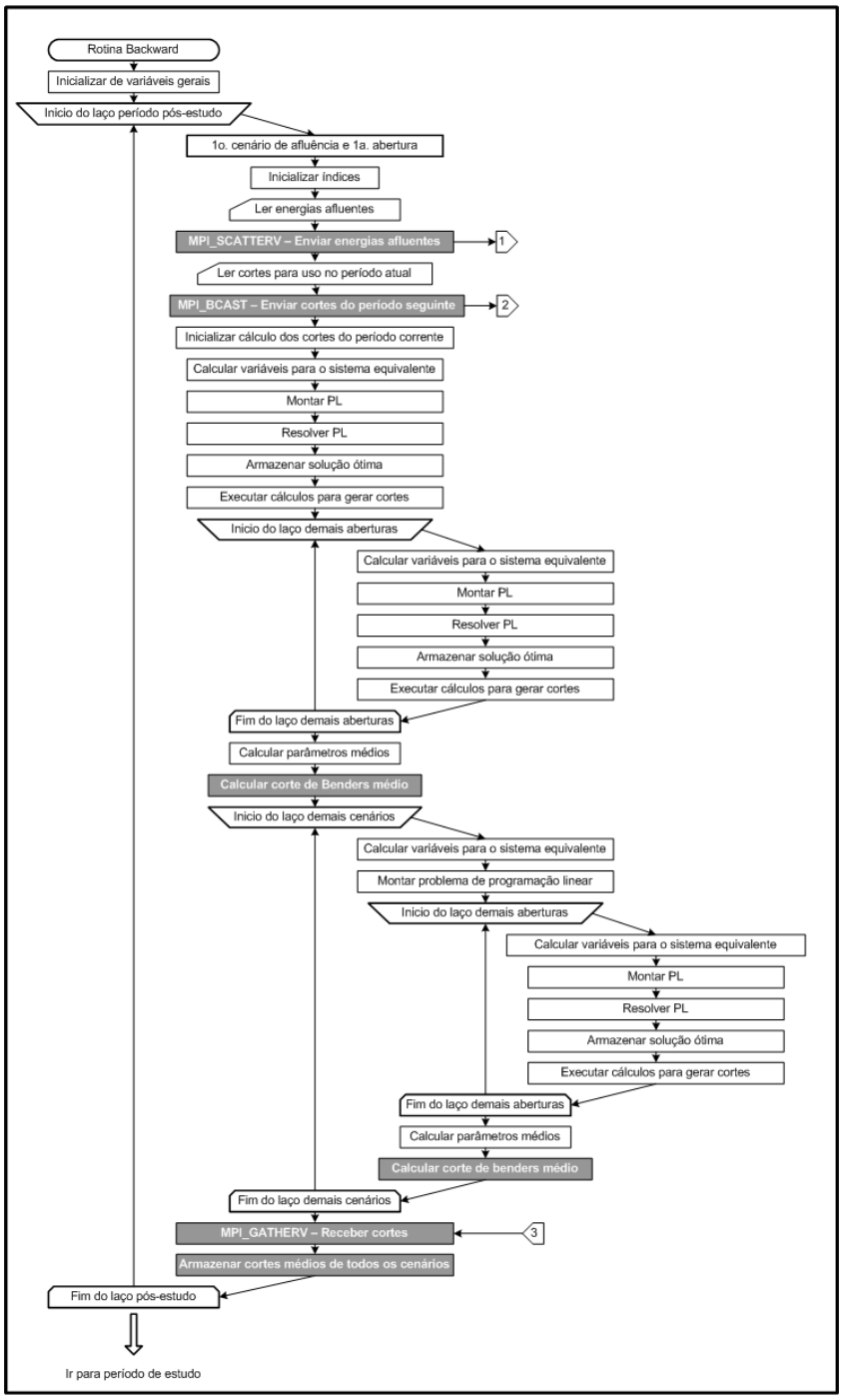

Figura 107 - Fluxograma da Rotina *Backward* (Versão Paralela – Processador Mestre)

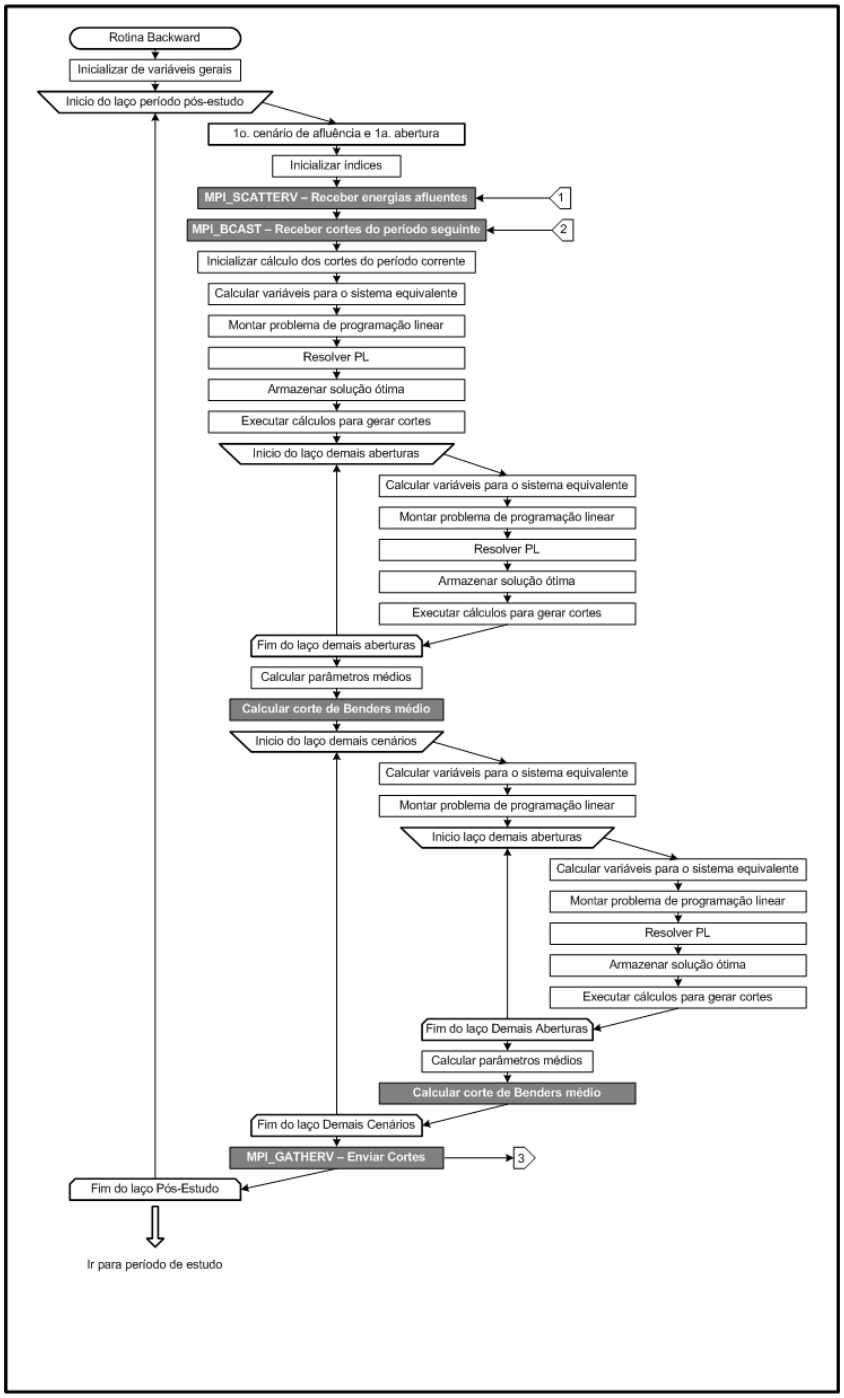

Figura 108 - Fluxograma da Rotina *Backward* (Versão Paralela – Demais Processadores)

- **Processo *Forward***

Nesse processo e execução das instruções começa pela primeira simulação do primeiro período. Tal qual no processo *backward*, o programa prepara e monta as

matrizes do primeiro PL. Depois desta solução, o programa resolve os PLs para os outros cenários. Estas instruções são repetidas para todos os outros períodos. Uma grande diferença ocorre no final de cada simulação onde são armazenados vários parâmetros para avaliação posterior da simulação.

Simplificadamente, as tarefas da rotina podem ser agrupadas nas seguintes etapas: (1) leitura de energias afluentes de períodos passados e do período atual; (2) leitura dos cortes para o período atual; (3) cálculos gerais; (4) inicialização do PL; (5) montagem das matrizes e solução do PL; (6) no final da primeira simulação do primeiro período ocorre o armazenamento da energia armazenada final para o próximo período, das variáveis para cálculo dos riscos anuais de déficit, das energias acumuladas e do custo de operação do estágio. Estas etapas são repetidas para as outras simulações dos outros períodos.

As modificações necessárias para permitir a execução distribuída dos cálculos desta rotina são as seguintes:

1. Concentrar todas as leituras no processador mestre. Este é o responsável pela transmissão dos dados lidos para todos os processadores. Para a transmissão dos dados das energias afluentes foram utilizadas funções *MPI_SCATTERV*, pois somente os dados referentes às simulações que serão calculadas pelo processador são significativos. Para a transmissão dos valores dos cortes de Benders do período corrente foram utilizadas funções *MPI_BCAST*;
2. Inicializar o valor da primeira simulação que será calculada pelo processador;
3. Ao final de cada simulação, armazenar em variáveis auxiliares as energias acumuladas e o limite superior da função objetivo;
4. Ao final de todas as simulações do período, transmitir os novos valores das energias armazenadas para o processador mestre. Nestas transmissões foram utilizadas funções *MPI_GATHERV*.

Ao final deste processo, é necessário transmitir as variáveis para cálculo de fatores de risco, da energia acumulada, do custo marginal de operação e da convergência para o processador mestre. Nestes casos, utilizaram-se as funções *MPI_GATHERV* e *MPI_REDUCE*. Esta última foi utilizada para as variáveis em que a soma dos valores de todas as simulações eram necessárias para o resultado final.

Na Figura 109 e na Figura 110 estão apresentados os fluxogramas da rotina *forward* com as instruções que são executadas pelo processador mestre e pelos demais processadores, respectivamente. Estes fluxogramas podem ser comparados com a versão seqüencial da rotina, mostrada na Figura 102. As diferenças entre as versões paralela e seqüencial estão mostradas através dos passos grifados em cinza.

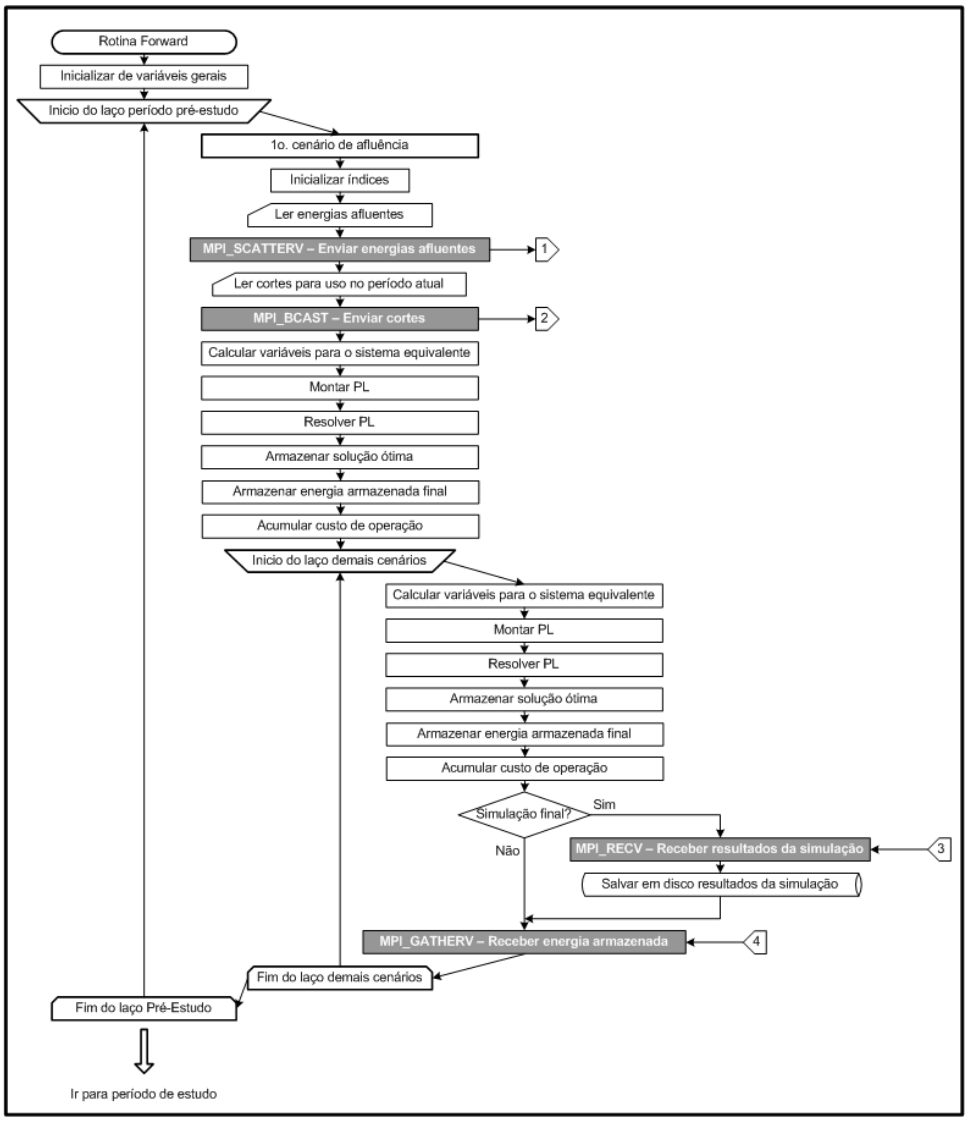

Figura 109 - Fluxograma da Rotina *Forward* (Versão Paralela – Processador Mestre)

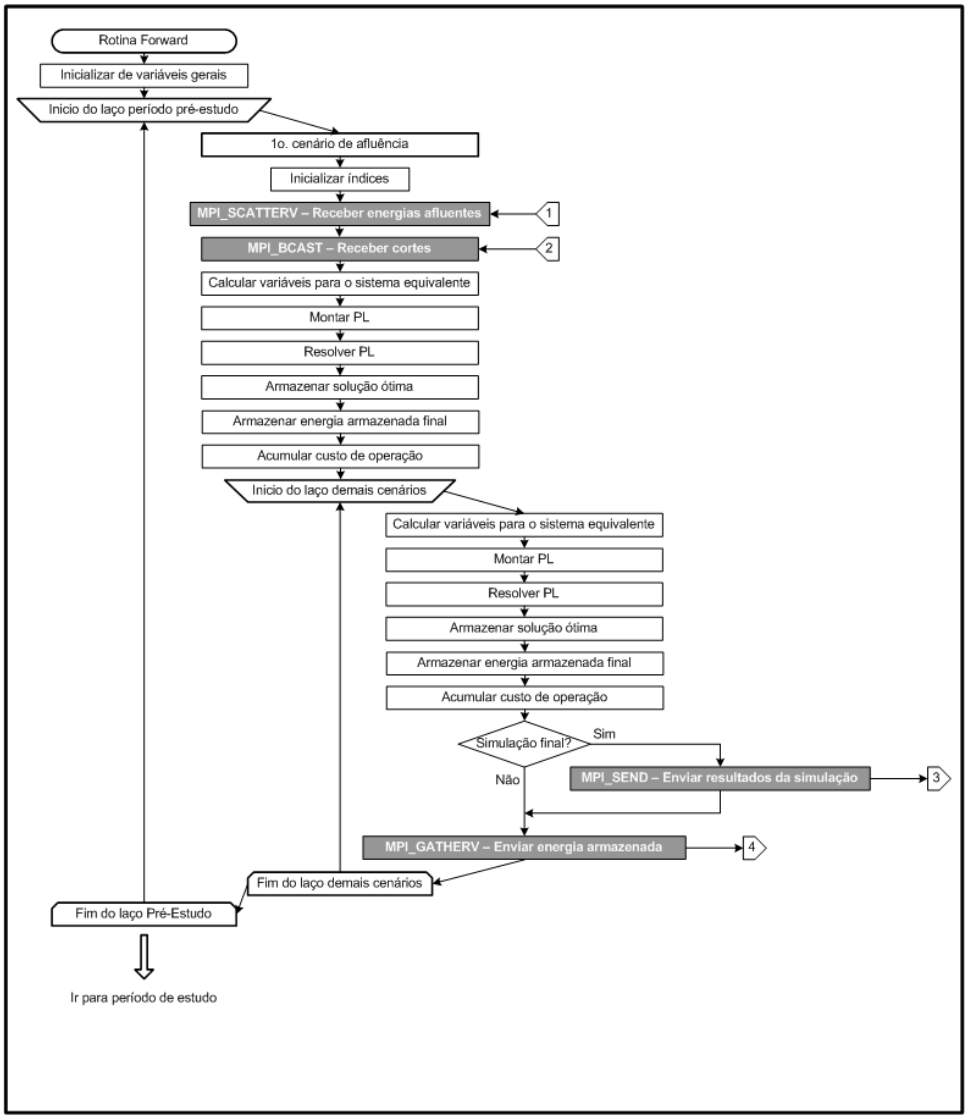

Figura 110 - Fluxograma da Rotina *Forward* (Versão Paralela – Demais Processadores)

- **Análise da Convergência**

Ao final da simulação *forward*, o programa verifica se houve a convergência. Esta verificação é feita no processador mestre, logo, todas as variáveis necessárias são transmitidas para este processador. Após essa verificação, ocorre o preenchimento de um sinalizador para indicar se houve convergência ou não e este sinalizador é enviado para todos os outros processadores através da função *MPI_BCAST*.

- **Envio dos Resultados de Todos os Processadores para o Processador Mestre**

A impressão dos resultados é similar ao procedimento adotado na leitura de dados, só que de forma inversa. Desta forma, ao final do processo, todos os resultados estarão concentrados no processador mestre. As instruções utilizadas foram a *MPI_GATHERV* e a *MPI_REDUCE.*

- **Finalização**

A finalização do ambiente do MPI é feita através da função *MPI_FINALIZE.*

A.4 Adequação do Código do Programa Newave

Neste item serão mostradas as modificações decorrentes do processamento paralelo em si, uma vez que algumas situações funcionam muito bem numa execução convencional, porém causam problemas na execução paralela.

- **Alteração do Modo de Armazenamento de Variáveis**

Durante a fase de estudo do código do programa Newave, verificou-se quais seriam os dados que deveriam ser transmitidos para cada processador. Alguns destes dados são variáveis simples, outras são vetores ou matrizes. O problema é que na maioria dos casos não se utilizam as dimensões máximas do programa, que são estáticas e definidas a priori no compilador Fortran. O problema mais crítico ocorre com as variáveis que tem a dimensão do número máximo de períodos, que normalmente é muito maior do que a quantidade utilizada na maioria dos estudos. Para não sobrecarregar as comunicações com o envio de valores desnecessários, decidiu-se transmitir somente a parte útil das variáveis que dependiam do número de períodos. Para que isso fosse possível, a dimensão referente a esse parâmetro deveria estar em último lugar em todas as variáveis envolvidas, permitindo a transmissão de pedaços da variável. Isso se deve ao modo de armazenamento por colunas do Fortran, como pode ser visto na Figura 111. No caso da variável `Matriz` o armazenamento se dá para uma unidade de dimensão $\texttt{Dim}_1$, mantendo-se as dimensões $\texttt{Dim}_2$ e $\texttt{Dim}_3$ fixas em 1. Depois mais uma unidade de dimensão $\texttt{Dim}_1$ é armazenada, só que com a dimensão $\texttt{Dim}_2$ incrementada de uma unidade e $\texttt{Dim}_3$ fixa em 1 e assim sucessivamente até que a

dimensão Dim_2 atinja o valor máximo. Neste ponto todos os dados referentes a uma unidade de dimensão Dim_3 estão armazenados. Os elementos armazenados a seguir correspondem a todo o processo anterior, só que para a segunda unidade da dimensão Dim_3 e assim sucessivamente até que todos os valores da variável `Matriz` estejam armazenados.

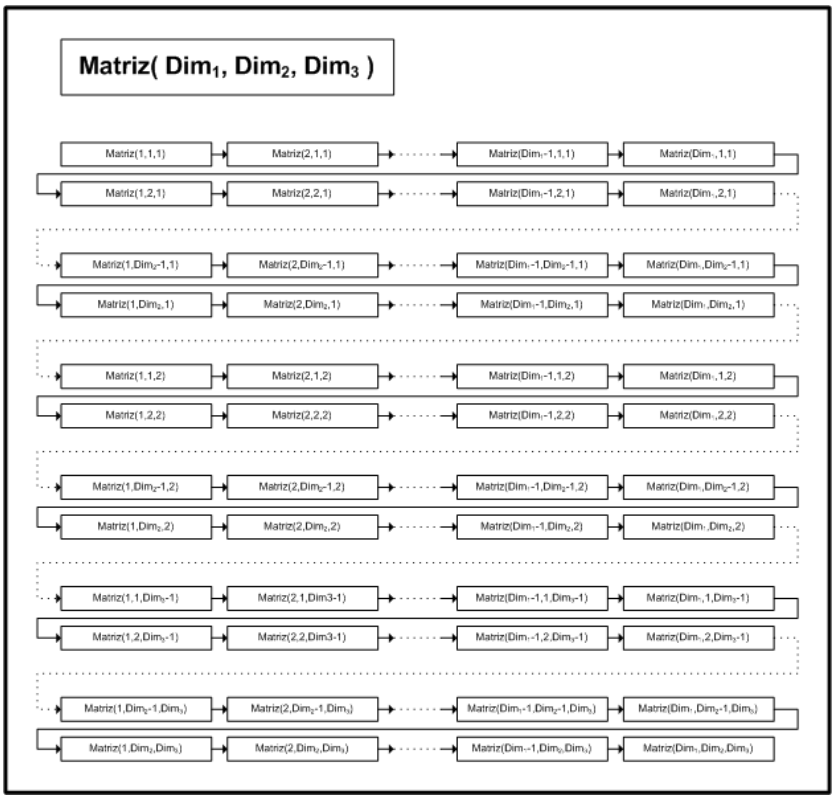

Figura 111 – Armazenamento de Matrizes em Compilador FORTRAN

Por causa deste modo de armazenamento, a dimensão de número de períodos tem que estar na mesma posição da dimensão Dim_3, ou seja, a última. Neste caso é possível separar pedaços da variável desejada, contendo todos os dados relacionados às outras dimensões para um subconjunto de dimensão Dim_3.

- **Variáveis a Serem Transmitidas**

Como foi adotado o procedimento de somente o processador mestre executar a leitura dos dados, foi necessário enviar para os outros processadores todos os dados necessários para que estes pudessem realizar as suas respectivas tarefas.

- **Alteração do Algoritmo de Solução do Programa Newave**

Durante a execução da versão com processamento paralelo do programa Newave, notou-se que, em alguns casos, aconteceram resultados distintos para um mesmo caso, quando executados com quantidades diferentes de processadores.

Após um exaustivo processo de depuração, descobriu-se que o algoritmo de solução dos PLs utilizado na versão sequencial e mantido originalmente na versão com processamento paralelo, não garante que, para o caso de múltiplas soluções, a solução seja sempre a mesma quando houver alteração da quantidade de processadores.

Dentro de um mesmo período, com o objetivo de acelerar a solução de um PL, excetuando-se o primeiro, a base ótima encontrada na solução do problema anterior é utilizada como ponto de partida. Logo, na execução com processamento paralelo, quando a quantidade de processadores é alterada, a base utilizada como ponto de partida para um PL pode mudar, principalmente se este PL passar a ser o primeiro a ser resolvido num outro processador. Caso este problema admita várias soluções, a mudança do ponto de partida pode levar à obtenção de diferentes resultados, todos corretos. Nestes casos, o valor final do custo de operação é o mesmo, porém com valores diferentes para outros parâmetros, como por exemplo, gerações dos subsistemas e, consequentemente, energias armazenadas finais diferentes. A partir das soluções diferentes deste PL, dois casos iguais podem seguir por trajetórias diferentes, resultando em soluções um pouco diferentes entre si, porém equivalentes sob o ponto de vista estatistico.

Mais especificamente, este problema pode ser mais bem entendido através das figuras a seguir. Na Figura 112 pode-se observar a solução sequencial de duas séries de um mesmo período, cada um com 20 cenários de energias afluentes (aberturas). No primeiro cenário da primeira série, o PL é resolvido e é obtida uma base viável, que é utilizada como ponto de partida para a solução do problema seguinte. Com isso, tem-se que o PL do segundo cenário da primeira série parte da base viável obtida no primeiro cenário. O mesmo ocorre para todos os outros cenários desta série, sempre utilizando a base obtida pelo problema imediatamente anterior. Quando o programa vai resolver o problema do primeiro cenário da série seguinte, a base da solução do último cenário da série anterior é utilizada como ponto de partida para a sua solução. O programa Newave

resolve os problemas de programação linear deste modo porque o ganho de tempo computacional é muito grande, uma vez que não é necessário obter uma base viável inicial para os problemas, exceto o primeiro.

Na Figura 113 tem-se a solução que foi adotada inicialmente para o processamento paralelo. Percebe-se claramente que ocorrerá uma diferença na execução paralela a partir da solução do primeiro problema de cada processador, uma vez que, enquanto que na solução convencional existia uma base viável inicial para ser aproveitada, na solução paralela esta base não existe, sendo necessária a sua determinação. Para problemas que só tenham uma solução, partir de bases viáveis diferentes não causa soluções diferentes, porém para um problema de múltiplas soluções pode-se obter soluções diferentes.

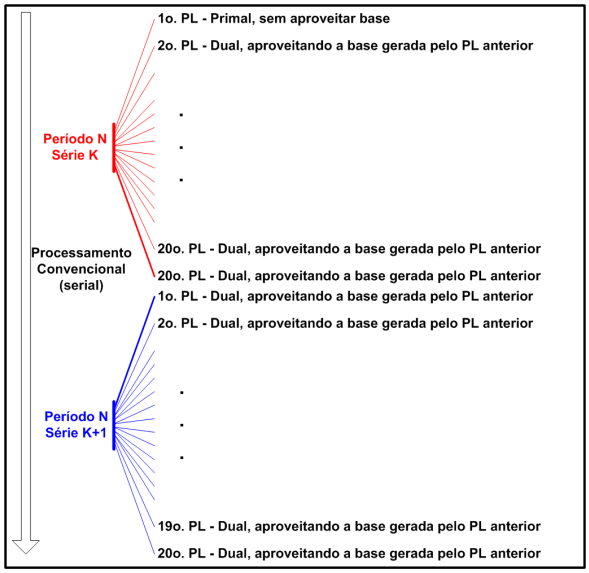

Figura 112 – Solução Convencional

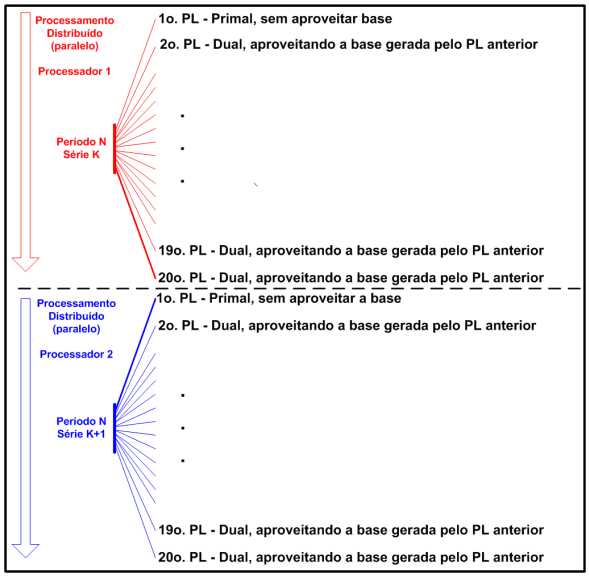

Figura 113 – Solução Inicial com Processamento Paralelo

A solução encontrada pode ser visualizada na Figura 114. O primeiro processador resolve o PL com o primeiro cenário de energia afluente, gerando uma base viável. Esta base é transmitida para todos os outros processadores participantes do ambiente de processamento paralelo, que estavam parados a espera da chegada desta base.

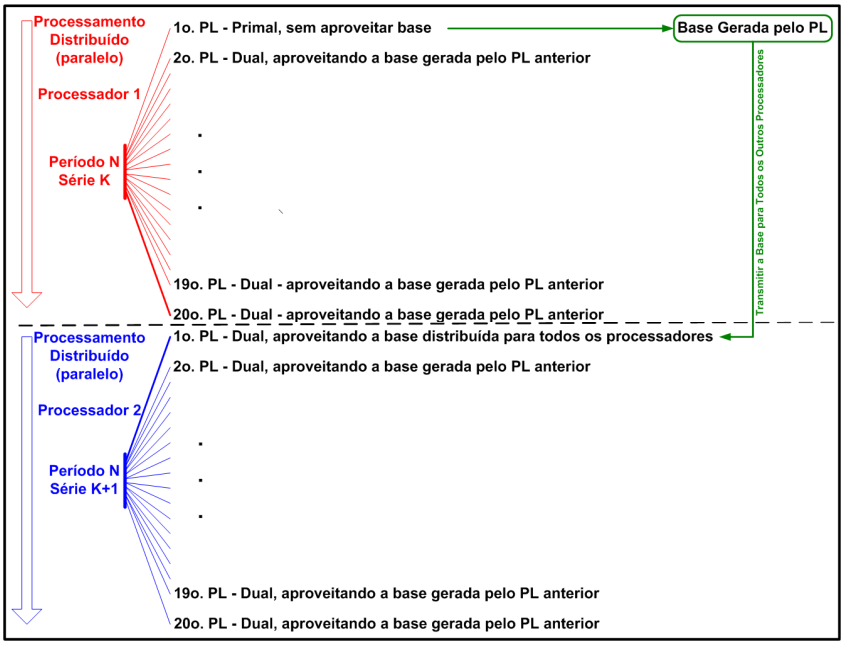

Figura 114 – Solução Final com Processamento Paralelo

Com este procedimento, pode-se garantir que o PL com múltiplas soluções será executado a partir de uma mesma base viável inicial, não importando a posição dele na solução de um determinado processador.

Esta alteração do algoritmo de solução do programa levou a alterações nos fluxogramas das rotinas *backward* e *forward* da versão com processamento paralelo, conforme pode ser observada nas quatro figuras a seguir.

Na Figura 115 e na Figura 116 estão apresentados os novos fluxogramas da rotina *backward*, para o processador mestre e para os demais, respectivamente. As diferenças em relação à solução paralela original, que estão grifadas em cinza, consistem na determinação de uma base viável pelo primeiro problema a ser resolvido pelo processador mestre, no envio desta base para todos os outros processadores e na utilização desta base em todos os outros problemas de um mesmo período.

Na Figura 117 e na Figura 118, o mesmo procedimento adotado na rotina *backward*, está apresentado para a rotina *forward*.

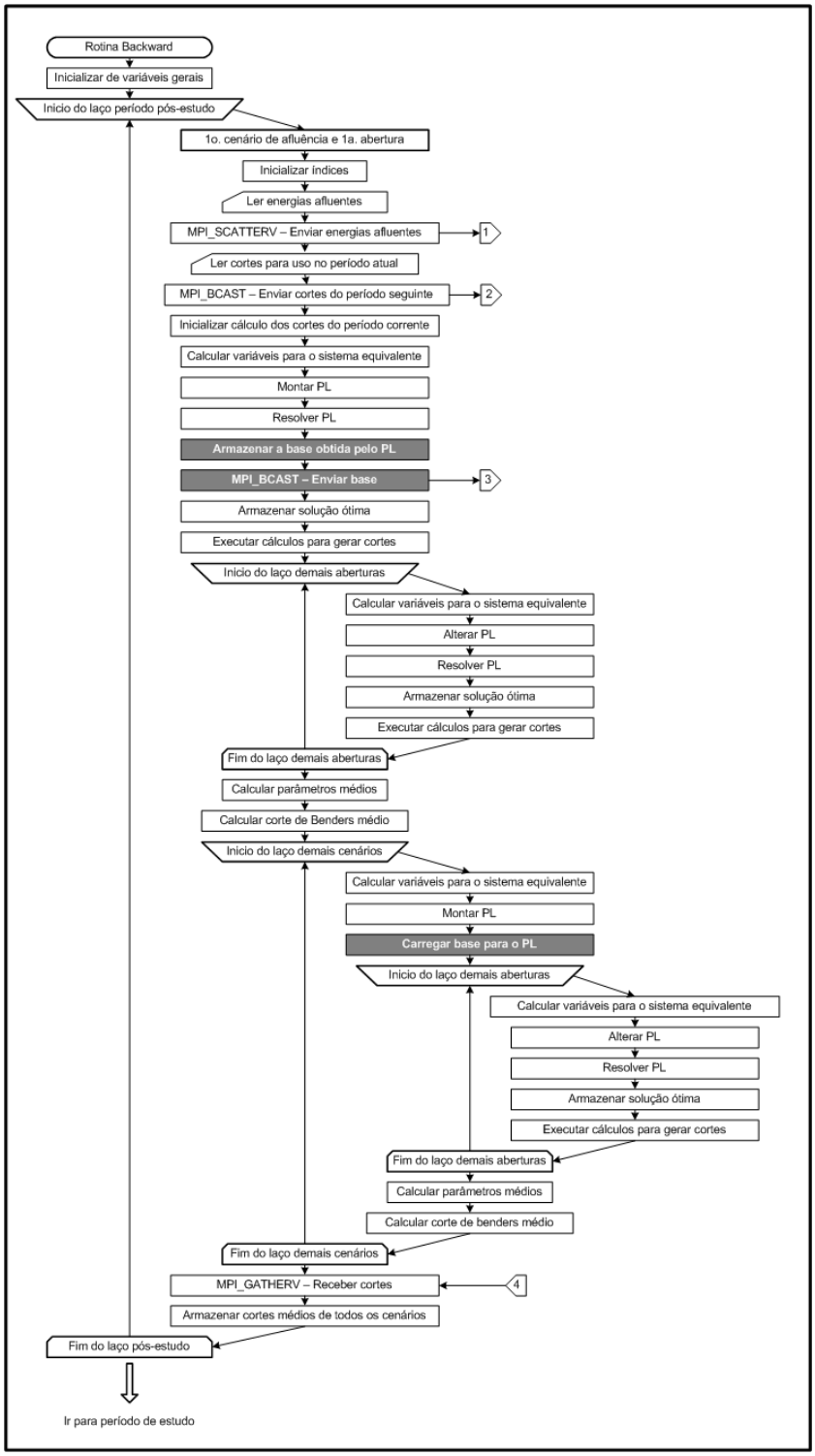

Figura 115 - Fluxograma da Rotina *Backward* com Envio de Base (Processador Mestre)

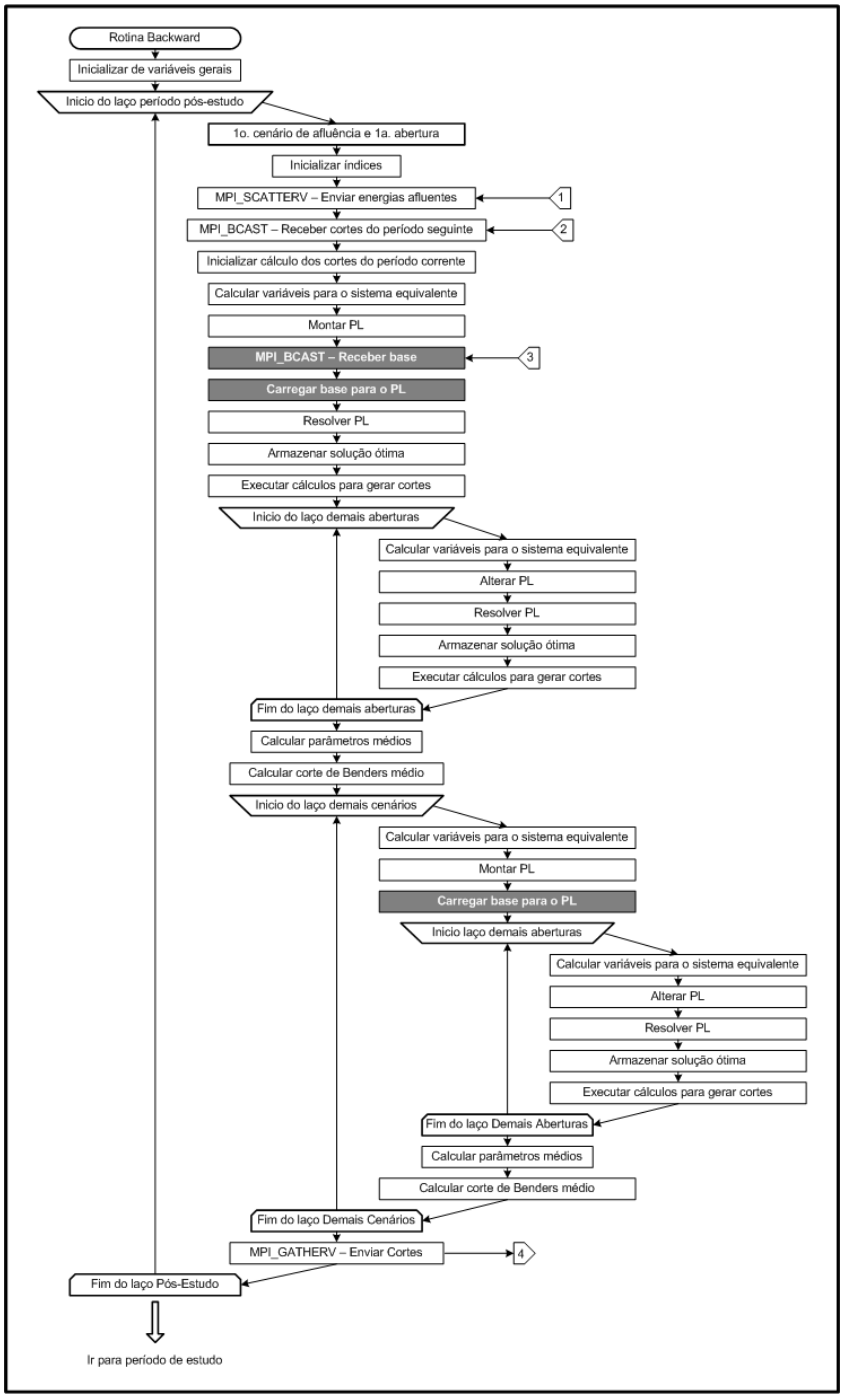

Figura 116 - Fluxograma da Rotina *Backward* com Envio de Base (Demais Processadores)

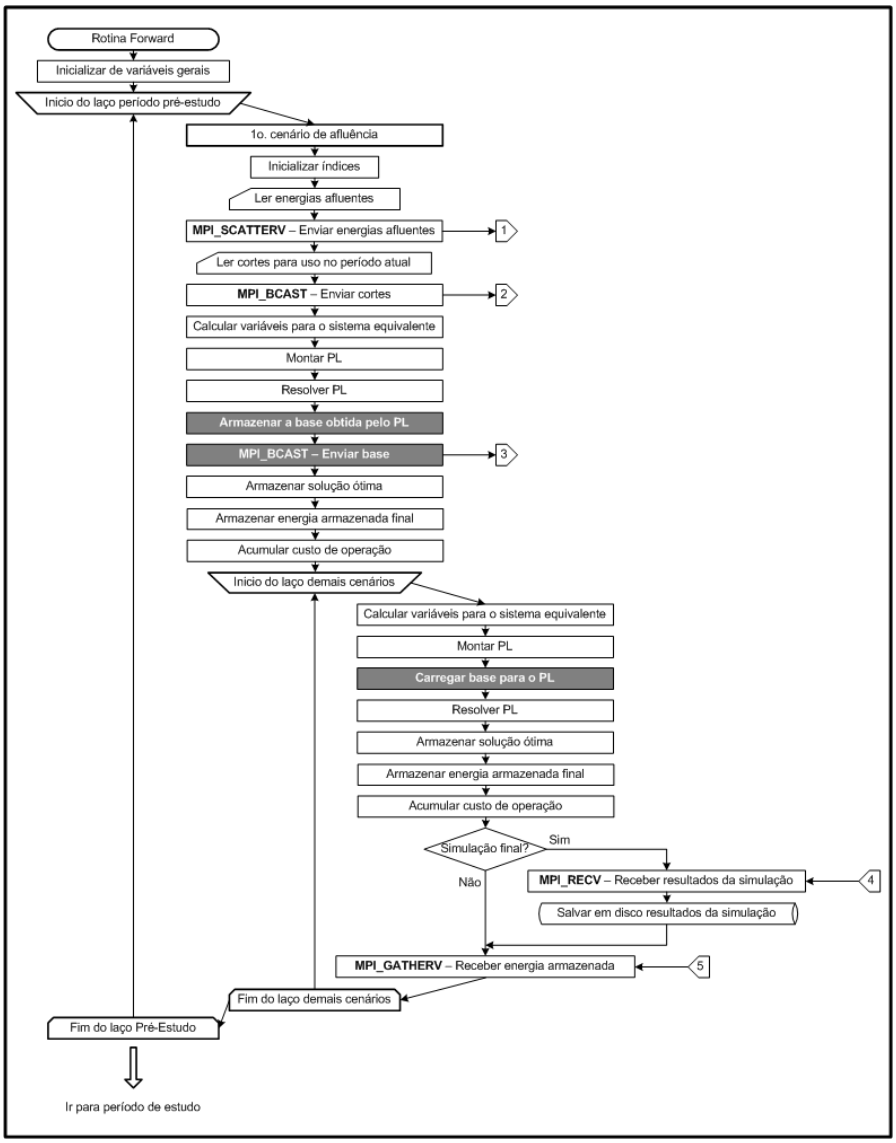

Figura 117 - Fluxograma da Rotina *Forward* com Envio de Base (Processador Mestre)

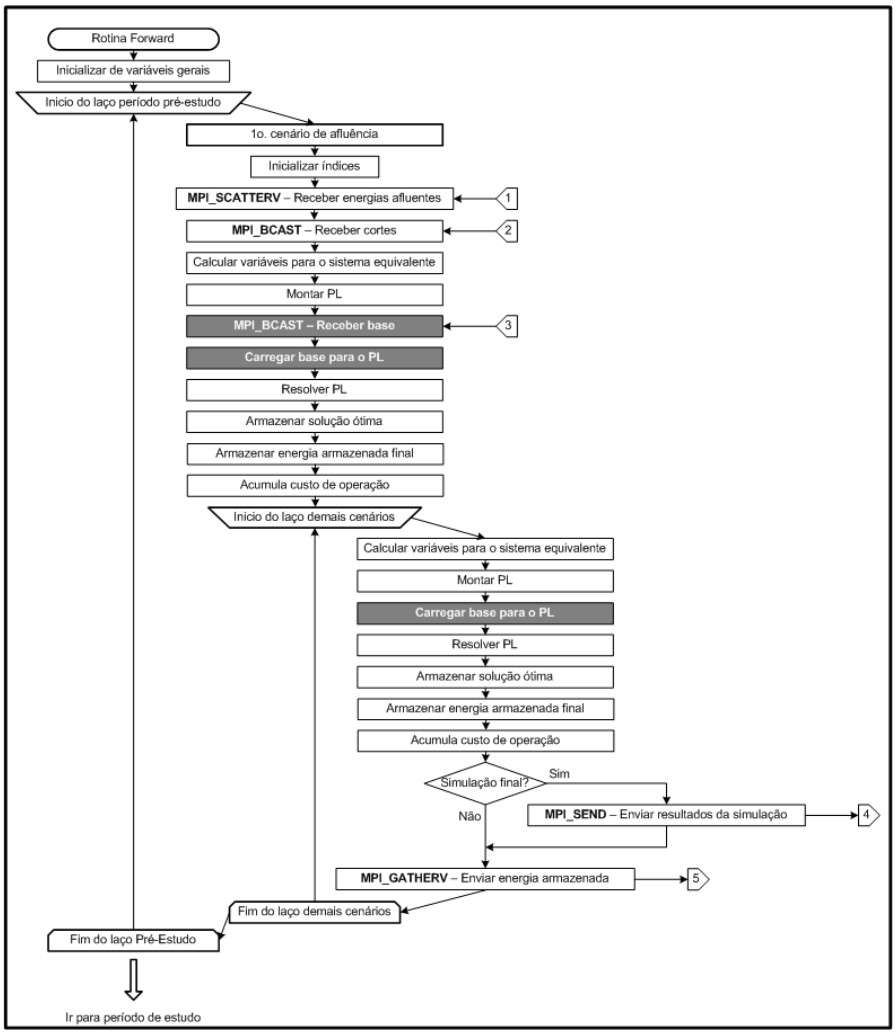

Figura 118 - Fluxograma da Rotina *Forward* com Envio de Base (Demais Processadores)

Anexo B

Detalhes da Otimização da Estratégia de Paralelização e da Obtenção de Resultados

Neste anexo estão apresentados alguns procedimentos adotados ao longo da análise para a otimização do desempenho da estratégia de paralelização.

B.1 Versão 2 - Otimização do Envio dos Cortes

Neste item estão apresentados os procedimentos necessários para se obter os tempos de processamento consumidos pelo programa de forma a permitir a identificação do problema do envio dos cortes e a sua solução.

- **Versão Inicialmente Proposta**

O primeiro passo para otimizar o desempenho da versão inicial é identificar o tempo gasto na execução das instruções do programa, de forma a se identificar onde o programa está consumindo mais tempo de processamento. Como seria inviável a colocação de instruções de tomada de tempo em cada linha do programa, decidiu-se por colocar algumas tomadas de tempo e determinar o tempo consumido de conjuntos de instruções. Desta forma, definiram-se os pontos que estão mostrados na Tabela 47.

Tabela 47 - Pontos de Tomada de Tempo na Rotina *backward* da Versão Inicial

Ponto	Processador	Descrição
0	Todos	Início do laço de períodos (referência)
1	Todos	Inicializações de variáveis
2	Mestre	Leitura dos cortes
	Demais	Nenhuma instrução executada
3	Todos	Envio de todos os cortes necessários no período atual. Este envio é feito através de uma instrução coletiva do processador mestre para todos os outros
4	Mestre	Leitura das energias afluentes de períodos passados e do período atual
	Demais	Nenhuma instrução executada

Ponto	Processador	Descrição
5	Todos	Envio das energias afluentes de períodos passados e do período atual. Este envio é feito através de uma instrução coletiva do processador mestre para todos os outros
6	Todos	Cálculos e montagem da matriz do problema da primeira abertura do primeiro período
7	Mestre	Solução do problema da primeira abertura do primeiro período
	Demais	Espera pela base viável do primeiro problema do processador Mestre
8	Mestre	Obtenção da base viável do problema da primeira abertura do primeiro problema
	Demais	Recebimento da base viável do problema da primeira abertura do primeiro período do processador Mestre
9	Mestre	Envio da base viável do problema da primeira abertura do primeiro período
	Demais	Solução do problema da primeira abertura do primeiro período
10	Todos	Armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda, armazenamento e dos cortes
11	Todos	Modificação do problema, solução e armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, das equações de balanço, demanda e armazenamento e dos cortes para todas as aberturas do primeiro período
12	Todos	Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos problemas de todas as aberturas dos demais períodos
13	Mestre	Recebimento de informações sobre a convergência do processo iterativo da rotina que recebe a solução ótima (envio via instrução ponto a ponto)
	Demais	Envio de informações sobre a convergência do processo iterativo da rotina que recebe a solução ótima (envio via instrução ponto a ponto)
14	Todos	Envio da primeira parte dos cortes de todos os processadores para o processador mestre
15	Todos	Envio da segunda parte dos cortes de todos os processadores para o processador mestre
16	Mestre	Cálculo e armazenamento dos cortes obtidos no período
	Demais	Nenhuma instrução executada
17	Todos	Fim do laço de períodos

Os resultados para cada um destes 17 pontos de tomada de tempo foram obtidos para cada um dos períodos da etapa de convergência do processo iterativo e para cada um dos processadores participantes do ambiente de processamento paralelo. O tempo consumido de cada ponto foi obtido através da diferença dos tempos reais medidos do ponto corrente e do anterior, de forma que o valor resultante fosse o tempo gasto entre o final do processamento das tarefas de dois pontos subsequentes. Logo, ao final do processamento do caso, 954.720 valores de tempos (17 pontos de tomada de tempo x 117 períodos x 32 processadores x 15 iterações) estavam disponíveis para análise.

Por conta da grande quantidade de valores, decidiu-se que, para facilitar a análise, todos os tempos de um mesmo ponto numa mesma iteração fossem somados.

Além disto, foram calculados os tempos médios obtidos nos processadores diferentes do mestre, para cada ponto de tomada de tempo e para cada iteração, conforme pode ser visto na Tabela 48. Destes valores podem-se traçar dois gráficos, um para os tempos do processador mestre e outro para os tempos médios dos demais processadores, conforme pode ser visto na Figura 119 e na Figura 120, mostradas a seguir.

Tabela 48 - Tempos de Cada Etapa do Ciclo *Backward* – Versão Inicial

Iter	Proc	Pontos de Tomada de Tempo (seg)																	Total
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
1	Mestre	0,00	0,10	0,64	0,23	0,46	0,28	0,79	0,00	0,11	0,02	4,84	32,57	21,64	0,03	0,00	1,09	0,00	62,79
	Demais	0,00	0,00	21,06	0,00	1,06	0,24	0,00	0,87	0,47	0,02	5,80	33,15	0,00	0,00	0,00	0,00	0,00	62,67
2	Mestre	0,00	0,28	14,70	0,20	0,42	0,31	1,06	0,00	0,15	0,03	8,54	57,31	6,98	0,03	0,00	2,64	0,00	92,64
	Demais	0,00	0,00	35,03	0,00	1,32	0,25	0,00	1,16	0,53	0,01	8,21	46,10	0,00	0,00	0,00	0,00	0,00	92,61
3	Mestre	0,00	0,45	29,24	0,22	0,29	0,43	1,38	0,00	0,11	0,02	10,27	72,57	8,76	0,03	0,00	4,20	0,00	127,98
	Demais	0,00	0,00	54,84	0,00	2,14	0,32	0,00	1,40	0,70	0,01	10,41	58,04	0,00	0,00	0,00	0,00	0,00	127,87
4	Mestre	0,00	0,64	43,47	0,26	0,32	0,40	1,90	0,00	0,14	0,02	15,22	95,02	12,49	0,03	0,01	5,77	0,00	175,68
	Demais	0,00	0,00	78,82	0,00	1,35	0,28	0,00	1,90	0,88	0,01	14,09	78,28	0,00	0,00	0,00	0,00	0,00	175,63
5	Mestre	0,00	0,82	55,40	0,26	0,34	0,43	2,47	0,00	1,61	0,02	15,13	109,30	14,37	0,03	0,01	7,31	0,00	207,50
	Demais	0,00	0,00	96,46	0,00	1,67	0,30	0,00	3,61	0,94	0,01	15,98	88,39	0,00	0,00	0,00	0,00	0,00	207,39
6	Mestre	0,00	0,99	64,73	0,26	0,35	0,46	2,98	0,00	2,58	0,03	20,66	139,34	16,61	0,03	0,01	8,88	0,00	257,90
	Demais	0,00	0,00	115,51	0,00	1,88	0,32	0,00	4,76	1,24	0,02	20,59	113,51	0,00	0,00	0,00	0,00	0,00	257,83
7	Mestre	0,00	1,18	68,65	0,27	0,34	0,49	3,47	0,00	4,97	0,03	23,39	152,40	19,88	0,03	0,00	10,44	0,00	285,53
	Demais	0,00	0,00	127,28	0,00	1,72	0,34	0,00	6,96	1,47	0,02	22,36	125,20	0,01	0,00	0,00	0,00	0,00	285,35
8	Mestre	0,00	1,36	75,07	0,40	0,34	0,53	3,96	0,01	5,66	0,02	25,03	177,63	21,15	0,03	0,01	12,00	0,00	323,17
	Demais	0,00	0,00	141,45	0,00	1,97	0,36	0,00	8,12	1,47	0,02	26,09	143,61	0,01	0,00	0,00	0,00	0,00	323,09
9	Mestre	0,00	1,55	82,89	0,25	0,33	0,58	4,54	0,00	5,57	0,02	28,81	194,49	24,91	0,02	0,01	13,55	0,00	357,51
	Demais	0,00	0,00	155,58	0,00	2,13	0,38	0,00	8,84	1,74	0,02	28,93	159,76	0,01	0,00	0,00	0,00	0,00	357,40
10	Mestre	0,00	1,72	86,47	0,24	0,32	0,60	5,28	0,01	6,57	0,03	32,64	221,26	21,15	0,03	0,00	15,13	0,00	391,44
	Demais	0,00	0,00	166,36	0,00	2,35	0,41	0,00	10,42	1,92	0,02	32,24	177,62	0,01	0,00	0,00	0,00	0,00	391,35
11	Mestre	0,00	1,91	94,48	0,25	0,34	0,64	5,87	0,01	7,25	0,02	34,35	240,10	31,25	0,03	0,01	16,65	0,00	433,14
	Demais	0,00	0,00	184,53	0,00	2,28	0,43	0,00	11,49	1,95	0,02	35,92	196,26	0,01	0,00	0,00	0,00	0,00	432,89
12	Mestre	0,00	2,09	101,35	0,25	0,32	0,68	6,85	0,01	6,89	0,02	38,33	259,66	33,09	0,04	0,00	18,22	0,00	467,79
	Demais	0,00	0,00	199,20	0,00	2,44	0,45	0,00	11,81	2,24	0,02	38,75	212,76	0,01	0,00	0,00	0,00	0,00	467,69
13	Mestre	0,00	2,27	103,25	0,29	0,34	0,72	7,48	0,01	8,66	0,03	43,01	285,49	35,34	0,03	0,00	19,78	0,00	506,70
	Demais	0,00	0,00	210,89	0,00	2,82	0,47	0,00	13,65	2,58	0,03	42,41	233,58	0,01	0,00	0,00	0,00	0,00	506,45
14	Mestre	0,00	2,45	109,59	0,27	0,35	0,77	8,46	0,01	9,43	0,03	45,15	301,48	38,22	0,03	0,00	21,44	0,00	537,68
	Demais	0,00	0,00	222,40	0,00	2,78	0,50	0,00	15,37	2,53	0,03	45,36	248,58	0,01	0,00	0,00	0,00	0,00	537,56
15	Mestre	0,00	2,63	114,94	0,27	0,34	0,81	9,45	0,01	9,60	0,03	49,94	331,11	40,48	0,03	0,00	22,98	0,00	582,61
	Demais	0,00	0,00	237,82	0,00	3,05	0,53	0,00	15,66	2,78	0,03	49,66	272,79	0,01	0,00	0,00	0,00	0,00	582,33

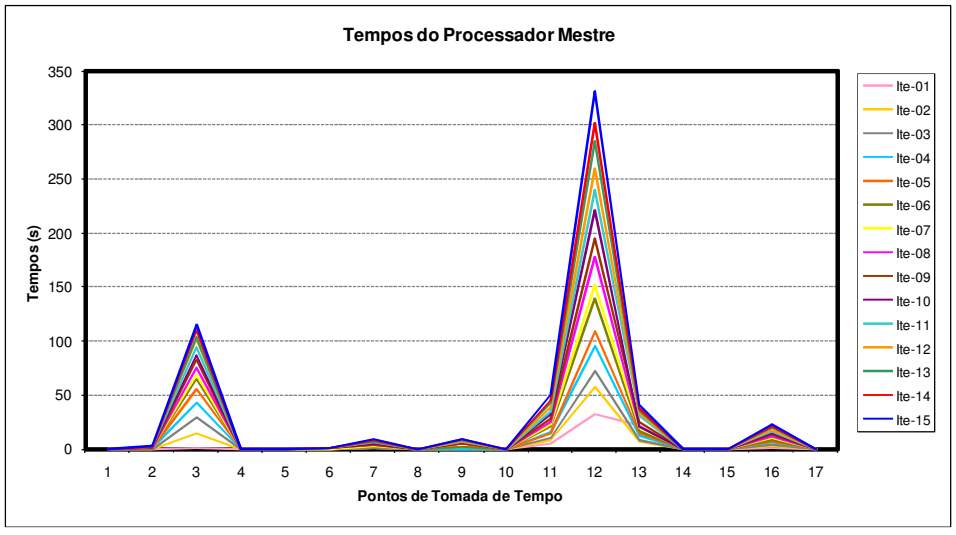

Figura 119 - Tempos de Processamento do Processador Mestre

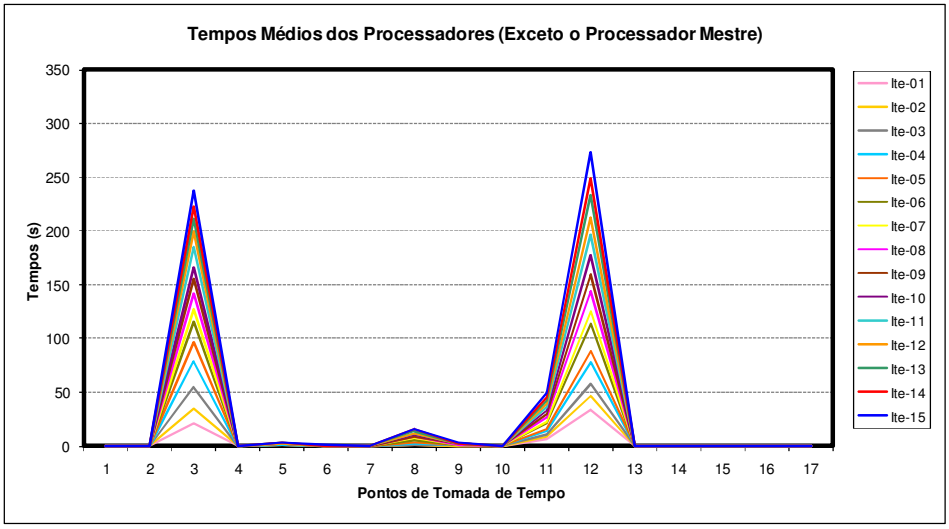

Figura 120 - Tempos Médios de Processamento dos Processadores Exceto o Mestre

Com relação ao processador mestre (Figura 119), os pontos em que os tempos de processamento foram significativos foram os seguintes:

- Ponto 3 – envio dos cortes para os demais processadores;
- Ponto 7 – solução do PL da primeira abertura do primeiro período;
- Ponto 9 – envio da base viável obtida na solução do PL da primeira abertura do primeiro período;
- Ponto 11 - Modificação do problema, solução e armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, das equações de balanço, demanda e armazenamento e dos cortes para todas as aberturas do primeiro período;
- Ponto 12 – Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos problemas de todas as aberturas dos demais períodos;
- Ponto 13 – Recebimento de informações sobre a convergência do processo iterativo da rotina que recebe a solução ótima;
- Ponto 16 – Cálculo e armazenamento dos cortes obtidos no período.

A análise dos tempos médios dos demais processadores (Figura 120), indica que os pontos com tempos significativos foram os seguintes:

- Ponto 3 – Recebimento dos cortes do processador mestre;
- Ponto 5 – Recebimento das energias afluentes de períodos passados e do período atual;
- Ponto 8 – Recebimento da base viável do problema da primeira abertura do primeiro período do processador mestre;
- Ponto 11 - Modificação do problema, solução e armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, das equações de balanço, demanda e armazenamento e dos cortes para todas as aberturas do primeiro período;
- Ponto 12 – Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos problemas de todas as aberturas dos demais períodos.

- **Versão 2**

Com a mudança na estratégia de envio/armazenamento dos cortes, algumas alterações nos pontos de tomada de tempo tiveram que ser alteradas, conforme está mostrado na Tabela 49, apresentada a seguir.

Tabela 49 - Pontos de Tomada de Tempo na Rotina *Backward* da versão com a Nova Estratégia no Tratamento dos Cortes

Ponto	Processador	Descrição
0	Todos	Início do laço de períodos (referência)
1	Todos	Inicializações de variáveis
2	Mestre Demais	Leitura das energias afluentes de períodos passados e do período atual Nenhuma instrução executada
3	Todos	Envio das energias afluentes de períodos passados e do período atual e informações da quantidade dos cortes que foram incluídos, eliminados e substituídos para o período atual (calculados no período anterior). Este envio é feito através de instruções coletivas do processador mestre para todos os outros
4	Mestre	Leitura dos cortes
	Demais	Leitura dos cortes armazenados nas iterações anteriores
5	Todos	Envio dos cortes incluídos no período atual da iteração corrente. Este envio é feito através de uma instrução coletiva do processador mestre para todos os outros.
6	Todos	Cálculos e montagem da matriz do problema da primeira abertura do primeiro período
7	Mestre	Solução do problema da primeira abertura do primeiro período
	Demais	Espera pela base viável do primeiro problema do processador Mestre
8	Mestre	Obtenção da base viável do problema da primeira abertura do primeiro problema

Ponto	Processador	Descrição
	Demais	Recebimento da base viável do problema da primeira abertura do primeiro período do processador Mestre
9	Mestre	Envio da base viável do problema da primeira abertura do primeiro período
	Demais	Solução do problema da primeira abertura do primeiro período
10	Todos	Armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes
11	Todos	Modificação do problema, solução e armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, das equações de balanço, demanda e armazenamento e dos cortes para todas as aberturas do primeiro período
12	Todos	Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos problemas de todas as aberturas dos demais períodos
13	Mestre	Recebimento de informações sobre a convergência do processo iterativo da rotina que recebe a solução ótima
	Demais	Envio de informações sobre a convergência do processo iterativo da rotina que recebe a solução ótima
14	Todos	Envio da primeira parte dos cortes de todos os processadores para o processador mestre
15	Todos	Envio da segunda parte dos cortes de todos os processadores para o processador mestre
16	Mestre	Cálculo e armazenamento dos cortes obtidos no período
	Demais	Armazenamento dos cortes obtidos no período anterior
17	Todos	Fim do laço de períodos

As diferenças dos pontos da versão original para esta segunda versão são as seguintes:

- Ponto 2 – Retorno da leitura das energias afluentes de períodos passados e do período atual no processador mestre para antes da leitura dos cortes (corresponde ao Ponto 4 da versão original);
- Ponto 3 – Envio das energias afluentes de períodos passados e do período atual do processador mestre para todos os demais, até aqui correspondia ao Ponto 5 da versão original, porém, aproveitou-se este ponto para transferir as informações da quantidade dos cortes que foram incluídos, eliminados e substituídos para o período atual;
- Ponto 4 – Para o processador mestre, este ponto é exatamente o mesmo do Ponto 2, porém, no caso dos demais processadores é feita a leitura dos cortes das iterações anteriores;

- Ponto 5 – Diferentemente do Ponto 3 da versão original, este ponto mede o tempo de processamento necessário para a transmissão dos cortes que foram inseridos apenas na iteração corrente e não todos os cortes da versão original;
- Ponto 16 – A diferença deste ponto em relação ao da versão original é que os demais processadores passam a salvar os cortes que foram enviados no Ponto 5 pelo processador mestre.

Os resultados da versão 2 estão apresentados na Tabela 50. Os novos tempos de processamento para o processador mestre e os valores médios dos demais processadores estão apresentados na Figura 121 e na Figura 122.

Tabela 50 - Tempos de Cada Etapa do Ciclo *Backward* – Versão 2

Iter	Proc	Pontos de Tomada de Tempo (seg)																	Total
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
1	Mestre	0,00	0,21	1,34	0,12	1,26	0,26	0,74	0,00	0,15	0,03	4,73	31,97	22,80	0,03	0,01	1,13	0,00	64,78
	Demais	0,00	0,00	21,51	0,00	1,87	0,23	0,00	1,36	0,47	0,01	5,82	33,20	0,00	0,00	0,00	0,16	0,00	64,65
2	Mestre	0,00	0,18	1,33	0,30	0,94	0,23	1,05	0,00	0,20	0,02	8,47	56,83	7,49	0,04	0,00	2,68	0,00	79,75
	Demais	0,00	0,00	20,97	0,10	1,49	0,26	0,00	1,78	0,53	0,01	8,21	46,14	0,00	0,00	0,00	0,22	0,00	79,72
3	Mestre	0,00	0,19	0,80	0,53	1,50	0,25	1,37	0,00	0,20	0,02	10,22	72,41	8,66	0,03	0,00	4,26	0,00	100,43
	Demais	0,00	0,00	26,84	0,21	1,38	0,31	0,00	2,23	0,71	0,01	10,41	58,03	0,00	0,00	0,00	0,22	0,00	100,34
4	Mestre	0,00	0,19	0,46	0,76	1,36	0,50	1,80	0,00	0,16	0,02	15,02	94,24	13,30	0,03	0,01	6,21	0,00	134,06
	Demais	0,00	0,00	35,98	0,31	0,95	0,59	0,00	2,65	0,89	0,01	14,10	78,30	0,00	0,00	0,00	0,22	0,00	134,02
5	Mestre	0,00	0,19	0,51	0,96	1,29	0,29	2,36	0,00	0,45	0,02	15,02	109,02	14,57	0,03	0,00	7,81	0,00	152,52
	Demais	0,00	0,00	41,65	0,41	1,03	0,37	0,00	3,40	0,95	0,01	15,98	88,39	0,01	0,00	0,00	0,22	0,00	152,41
6	Mestre	0,00	0,19	0,58	1,12	1,23	0,30	2,84	0,00	0,50	0,02	20,45	138,33	17,62	0,03	0,00	9,03	0,00	192,24
	Demais	0,00	0,00	50,69	0,50	1,12	0,37	0,00	3,95	1,24	0,02	20,58	113,49	0,00	0,00	0,00	0,23	0,00	192,18
7	Mestre	0,00	0,20	0,54	1,33	1,19	0,33	3,29	0,00	0,71	0,02	23,26	151,92	20,15	0,03	0,00	10,46	0,00	213,45
	Demais	0,00	0,00	57,13	0,60	1,19	0,41	0,00	4,56	1,47	0,02	22,38	125,30	0,01	0,00	0,00	0,22	0,00	213,28
8	Mestre	0,00	0,19	0,62	1,51	1,20	0,33	3,73	0,00	0,80	0,02	24,86	176,53	21,87	0,04	0,00	12,04	0,00	243,72
	Demais	0,00	0,00	64,91	0,69	1,25	0,42	0,00	4,93	1,48	0,02	26,09	143,61	0,01	0,00	0,00	0,22	0,00	243,63
9	Mestre	0,00	0,19	0,62	1,68	1,10	0,38	4,31	0,01	0,86	0,02	28,80	194,65	24,35	0,03	0,01	13,79	0,00	270,78
	Demais	0,00	0,00	72,02	0,78	1,28	0,45	0,00	5,54	1,72	0,02	28,90	159,66	0,01	0,00	0,00	0,22	0,00	270,61
10	Mestre	0,00	0,19	0,64	1,87	1,15	0,38	5,12	0,00	0,91	0,02	32,34	219,29	22,75	0,03	0,01	15,28	0,00	299,97
	Demais	0,00	0,00	78,81	0,88	1,40	0,45	0,00	6,41	1,93	0,02	32,22	177,52	0,00	0,00	0,00	0,21	0,00	299,87
11	Mestre	0,00	0,19	0,56	2,08	1,02	0,42	5,58	0,01	0,88	0,03	34,23	239,89	31,43	0,03	0,00	16,66	0,00	333,02
	Demais	0,00	0,00	88,49	0,98	1,39	0,47	0,00	6,84	1,97	0,02	35,94	196,44	0,01	0,00	0,00	0,21	0,00	332,78
12	Mestre	0,00	0,19	0,59	2,27	1,00	0,47	6,55	0,01	0,92	0,02	38,17	258,52	33,91	0,04	0,01	18,31	0,00	360,97
	Demais	0,00	0,00	95,96	1,07	1,50	0,49	0,00	7,89	2,26	0,03	38,73	212,70	0,01	0,00	0,00	0,22	0,00	360,86
13	Mestre	0,00	0,19	0,54	2,48	1,16	0,51	7,17	0,00	0,96	0,03	43,06	285,74	34,49	0,04	0,00	19,79	0,00	396,16
	Demais	0,00	0,00	105,33	1,17	1,76	0,51	0,03	8,49	2,60	0,03	42,38	233,41	0,01	0,00	0,00	0,22	0,00	395,94
14	Mestre	0,00	0,19	0,57	2,65	1,40	0,55	8,09	0,01	0,96	0,03	44,94	300,59	38,64	0,03	0,01	21,42	0,00	420,06
	Demais	0,00	0,00	109,90	1,26	2,08	0,54	0,00	9,48	2,55	0,03	45,34	248,53	0,01	0,00	0,00	0,22	0,00	419,94
15	Mestre	0,00	0,20	0,52	2,84	1,38	1,55	9,02	0,01	1,00	0,04	49,72	330,09	40,78	0,03	0,01	22,99	0,00	460,17
	Demais	0,00	0,00	119,14	1,35	2,14	0,59	0,00	10,40	2,80	0,03	49,62	272,62	0,01	0,00	0,00	0,22	0,00	458,91

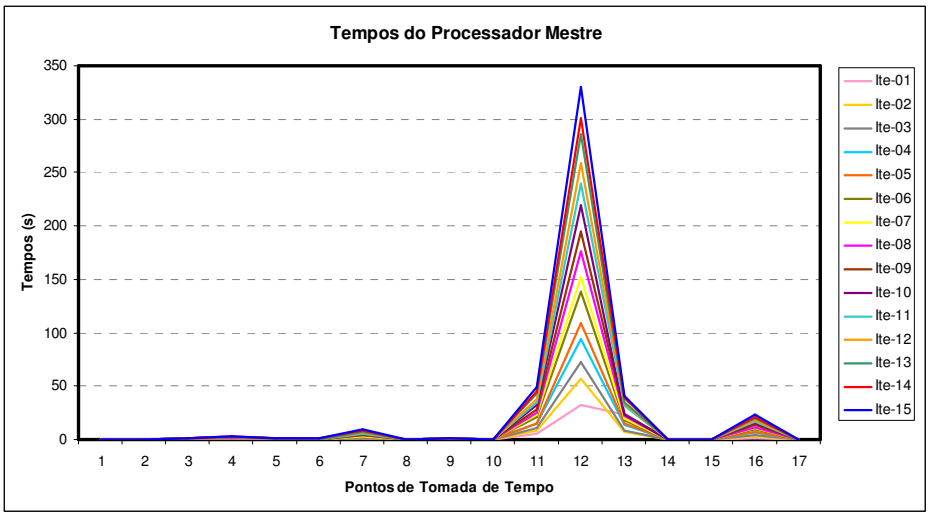

Figura 121 - Tempos de Processamento (s) do Processador Mestre (Nova Estratégia no Envio/Armazenamento dos Cortes)

Na Figura 121, os pontos em que os tempos de processamento foram significativos foram os seguintes:

- Ponto 7 – solução do PL da primeira abertura do primeiro período;
- Ponto 11 – Modificação do problema, solução e armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, das equações de balanço, demanda e armazenamento e dos cortes para todas as aberturas do primeiro período;
- Ponto 12 – Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos problemas de todas as aberturas dos demais períodos;
- Ponto 13 – Recebimento de informações sobre a convergência do processo iterativo da rotina que recebe a solução ótima;
- Ponto 16 – Cálculo e armazenamento dos cortes obtidos no período.

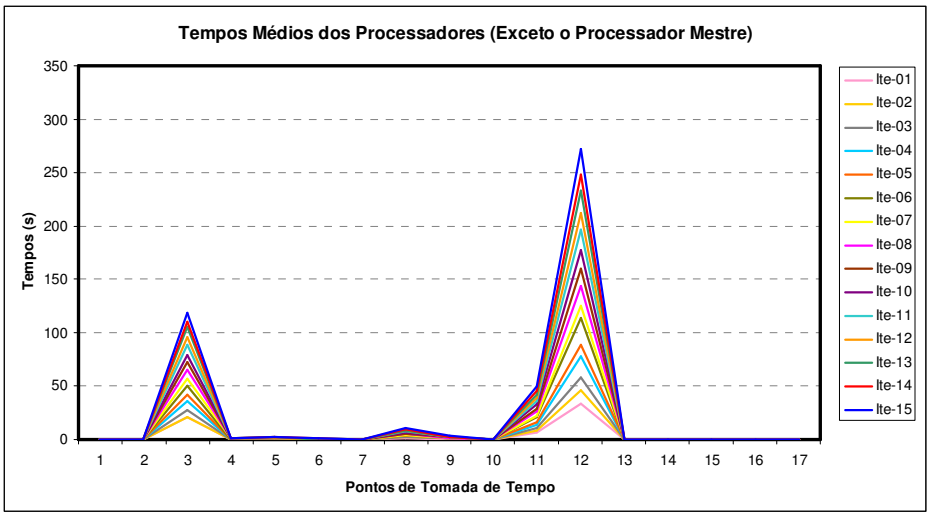

Figura 122 - Tempos Médios de Processamento (s) dos Processadores Exceto o Mestre (Versão com Nova Estratégia no Envio/Armazenamento dos Cortes)

Na Figura 122, os pontos com tempos de processamento significativos foram os seguintes:

- Ponto 3 – Recebimento das energias afluentes de períodos passados e do período atual e das informações da quantidade dos cortes que foram incluídos, eliminados e substituídos para o período atual;
- Ponto 8 – Recebimento da base viável do problema da primeira abertura do primeiro período do processador mestre;
- Ponto 11 - Modificação do problema, solução e armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, das equações de balanço, demanda e armazenamento e dos cortes para todas as aberturas do primeiro período;
- Ponto 12 – Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos problemas de todas as aberturas dos demais períodos.

B.2 Versão 3 - Balanceamento Dinâmico de Carga dos Processadores

A seguir está descrita como foi feita a identificação das diferenças dos tempos de processamento para a execução de tarefas similares entre os processadores. Neste processo, os pontos de tomada de tempo foram os mesmos adotados no item anterior.

De acordo com os valores mostrados na Tabela 51, os tempos referentes ao ponto 11, que consiste na solução dos PLs das demais aberturas da primeira série hidrológica, correspondente à solução de 19 PLs em cada período, apresentam uma variação de 12,13s, limites de 43,34s (processador 17) e 55,47s (processador 13). Com relação ao ponto 12, que consiste na solução dos PLs das demais series hidrológicas e suas respectivas aberturas, a variação dos tempos foi de 83,27s, causados por um mínimo de 246,82s (processador 14) e um máximo de 330,09s (processador mestre).

Tabela 51 - Somatórios dos Tempos de Cada Etapa do Ciclo *Backward* – 15ª Iteração – Versão 2

Ciclo Backward - Tempos de Cada Etapa (s)																		
Proc	Pto. 1	Pto. 2	Pto. 3	Pto. 4	Pto. 5	Pto. 6	Pto. 7	Pto. 8	Pto. 9	Pto. 10	Pto. 11	Pto. 12	Pto. 13	Pto. 14	Pto. 15	Pto. 16	Pto. 17	Total
1	0,00	0,20	0,52	2,84	1,38	1,55	9,02	0,01	1,00	0,04	49,72	330,09	40,78	0,03	0,01	22,99	0,00	460,17
2	0,00	0,00	77,80	1,37	2,15	0,55	0,00	10,32	2,52	0,04	49,36	314,55	0,01	0,00	0,00	0,22	0,00	458,88
3	0,00	0,00	88,82	1,33	1,94	0,58	0,00	10,35	2,98	0,03	47,66	305,15	0,01	0,00	0,00	0,21	0,00	459,07
4	0,00	0,00	82,02	1,55	2,14	0,63	0,00	10,44	3,30	0,03	48,07	310,61	0,01	0,00	0,00	0,21	0,00	459,00
5	0,00	0,00	82,83	1,35	3,13	0,57	0,00	10,21	2,67	0,03	45,52	312,47	0,01	0,00	0,00	0,22	0,00	459,00
6	0,00	0,00	80,47	1,52	2,14	0,54	0,00	10,43	2,71	0,05	52,24	308,76	0,01	0,00	0,00	0,22	0,00	459,08
7	0,00	0,00	91,03	1,57	1,95	0,59	0,00	10,45	2,73	0,02	49,47	300,97	0,00	0,00	0,00	0,21	0,00	459,00
8	0,00	0,00	85,18	1,40	2,06	0,60	0,00	10,58	3,13	0,03	46,82	308,92	0,00	0,00	0,00	0,22	0,00	458,94
9	0,00	0,00	116,40	1,36	3,10	0,57	0,00	10,13	2,66	0,03	48,51	275,96	0,01	0,00	0,00	0,22	0,00	458,93
10	0,00	0,00	112,95	1,40	2,13	0,57	0,00	10,37	2,45	0,04	49,83	279,05	0,00	0,00	0,00	0,26	0,00	459,05
11	0,00	0,00	130,98	1,38	2,04	0,63	0,00	10,39	2,92	0,02	50,83	259,69	0,01	0,00	0,00	0,21	0,00	459,09
12	0,00	0,00	129,32	1,40	2,02	0,65	0,00	10,54	3,09	0,03	51,65	260,02	0,01	0,00	0,00	0,24	0,00	458,96
13	0,00	0,00	128,26	1,32	3,01	0,57	0,00	10,41	2,90	0,06	55,47	256,75	0,01	0,00	0,00	0,22	0,00	458,99
14	0,00	0,00	143,53	1,34	1,98	0,55	0,00	10,65	2,65	0,03	50,84	246,82	0,00	0,00	0,00	0,28	0,00	458,67
15	0,00	0,00	128,19	1,34	1,83	0,58	0,00	10,64	2,70	0,02	46,01	267,12	0,01	0,00	0,00	0,21	0,00	458,67
16	0,00	0,00	116,70	1,30	1,90	0,61	0,00	10,76	2,39	0,03	51,68	273,28	0,00	0,00	0,00	0,21	0,00	458,87
17	0,00	0,00	148,75	1,36	3,06	0,57	0,00	9,88	2,62	0,03	43,34	248,99	0,01	0,00	0,00	0,26	0,00	458,86
18	0,00	0,00	142,62	1,30	1,96	0,57	0,00	10,12	3,06	0,03	49,42	249,58	0,00	0,00	0,00	0,22	0,00	458,88
19	0,00	0,00	135,44	1,31	1,82	0,57	0,00	10,16	2,75	0,03	49,08	257,44	0,01	0,01	0,00	0,21	0,00	458,83
20	0,00	0,00	128,06	1,33	1,80	0,63	0,00	10,28	2,74	0,02	49,35	264,41	0,00	0,00	0,00	0,22	0,00	458,84
21	0,00	0,00	119,26	1,30	2,88	0,57	0,00	10,17	2,94	0,03	48,15	273,65	0,01	0,00	0,00	0,21	0,00	459,16
22	0,00	0,00	138,24	1,29	1,86	0,55	0,00	10,37	2,81	0,03	45,97	257,55	0,01	0,00	0,00	0,25	0,00	458,93
23	0,00	0,00	138,73	1,29	1,75	0,60	0,00	10,40	2,91	0,06	52,10	250,75	0,01	0,01	0,00	0,23	0,00	458,83
24	0,00	0,00	130,26	1,37	1,77	0,60	0,00	10,54	2,93	0,03	48,26	262,71	0,01	0,00	0,00	0,23	0,00	458,69
25	0,00	0,00	127,25	1,30	2,82	0,59	0,00	10,11	2,84	0,03	52,29	261,46	0,01	0,00	0,00	0,21	0,00	458,92
26	0,00	0,00	116,52	1,28	1,86	0,56	0,00	10,36	2,70	0,04	54,17	271,15	0,01	0,00	0,00	0,21	0,00	458,85
27	0,00	0,00	125,67	1,33	1,71	0,60	0,00	10,34	2,81	0,02	47,03	268,93	0,00	0,01	0,00	0,24	0,00	458,71
28	0,00	0,00	132,45	1,28	1,82	0,62	0,00	10,53	2,42	0,02	46,89	262,52	0,01	0,00	0,00	0,22	0,00	458,77
29	0,00	0,00	137,24	1,30	2,67	0,62	0,00	10,38	2,88	0,04	50,23	253,42	0,01	0,00	0,00	0,21	0,00	459,00
30	0,00	0,00	119,64	1,30	1,70	0,54	0,00	10,64	2,71	0,03	53,12	269,11	0,00	0,00	0,00	0,22	0,00	459,02
31	0,00	0,00	128,87	1,28	1,65	0,59	0,00	10,67	2,84	0,02	53,35	259,51	0,01	0,00	0,00	0,21	0,00	459,00
32	0,00	0,00	129,78	1,30	1,67	0,60	0,00	10,83	2,94	0,02	51,47	259,84	0,00	0,00	0,00	0,22	0,00	458,69

Estas diferenças mostram que o balanceamento de carga estático não é igualitário, fazendo com que ocorram tempos significativos de ociosidade nos processadores, prejudicando a eficiência da estratégia de paralelização.

B.3 Versão 6 - Compensação da Deficiência do *Hardware*

Neste item está descrita como foi feita a identificação da diferença de tempo de processamento entre as execuções das versões 5 e 6 da estratégia de paralelização.

Como a versão atual da estratégia difere daquela inicial onde os pontos de tomada de tempo foram definidos, algumas tomadas de tempos foram colocadas em pontos diferentes, totalizando 13 pontos ao longo da rotina *backward*. Destes pontos, é importante destacar os seguintes:

- Ponto 4 → Envio/recebimento dos cortes produzidos na iteração anterior;
- Ponto 7 → Cálculos de parâmetros, montagem da matriz do problema e solução do primeiro PL para gerar a base viável para os demais PLs;
- Ponto 9 → Cálculos e montagem da matriz do problema, solução, armazenamento da solução ótima, cálculo do PI da água, acumulação dos valores da função objetivo, equações de balanço, demanda e armazenamento e dos cortes dos PLs para todas as aberturas da série hidrológica;
- Ponto 12 → Envio/recebimento dos cortes da iteração corrente;
- Ponto 13 → Enquanto o processador mestre calcula e armazenamento dos cortes obtidos no período corrente, os demais processadores armazena os cortes obtidos no período calculado imediatamente anterior.

Com a definição dos pontos de medição de tempo, após a execução dos casos obteve-se uma quantidade muito grande de valores, tornando necessária a escolha de um trecho de valores para realizar a análise. Optou-se por escolher o período, da última iteração, em que os tempos de solução dos PLs das aberturas fossem máximos. Estes valores ocorreram no período 80 da 28ª iteração, conforme estão apresentados na Tabela 52 e na Tabela 53, mostradas a seguir.

Tabela 52 – Tempos de Processamento (s) do Caso PMO Março/2009 com 50 Aberturas – Versão 5

Caso PMO Março/2009 – 200 Séries – 50 Aberturas – 16 Processadores – Versão 5																		
Iter.	Per.	Pto.	Proc. 1	Proc. 2	Proc. 3	Proc. 4	Proc. 5	Proc. 6	Proc. 7	Proc. 8	Proc. 9	Proc. 10	Proc. 11	Proc. 12	Proc. 13	Proc. 14	Proc. 15	Proc. 16
28	80	1	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	3	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	4	0,00	2,81	1,54	0,60	3,06	2,48	3,57	0,46	3,53	3,64	2,65	1,45	1,02	3,59	2,30	2,97
28	80	5	0,05	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03	0,03
28	80	6	0,00	0,02	0,02	0,02	0,02	0,02	0,02	0,01	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02
28	80	7	0,26	0,26	0,25	0,25	0,25	0,26	0,26	0,25	0,26	0,26	0,26	0,26	0,25	0,26	0,26	0,25
28	80	8	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00	0,00	0,01	0,00
28	80	9	40,63	40,85	38,38	39,84	39,10	38,42	38,43	41,25	39,13	39,40	40,34	38,29	39,31	38,93	38,66	40,71
28	80	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	11	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	12	0,61	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	13	0,41	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	1	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	3	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	4	0,01	0,81	3,29	1,83	2,57	3,24	3,23	0,42	2,52	2,27	1,32	3,37	2,36	2,73	3,00	0,96
		1-13	41,97	43,96	40,23	40,74	42,46	41,21	42,30	42,00	42,98	43,34	43,30	40,05	40,63	42,82	41,27	43,98
		5-4	41,98	41,97	41,98	41,97	41,97	41,96	41,96	41,96	41,97	41,97	41,97	41,97	41,97	41,97	41,97	41,97

Tabela 53 - Tempos de Processamento (s) do Caso PMO Março/2009 com 50 Aberturas – Versão 6

Caso PMO Março/2009 – 200 Séries – 50 Aberturas – 16 Processadores – Versão 6																		
Iter.	Per.	Pto.	Proc. 1	Proc. 2	Proc. 3	Proc. 4	Proc. 5	Proc. 6	Proc. 7	Proc. 8	Proc. 9	Proc. 10	Proc. 11	Proc. 12	Proc. 13	Proc. 14	Proc. 15	Proc. 16
28	80	1	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	3	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	4	0,00	3,68	3,81	3,73	3,68	1,75	2,75	2,84	3,14	2,50	0,45	1,66	0,89	3,42	3,12	3,17
28	80	5	0,04	0,02	0,02	0,03	0,02	0,03	0,03	0,02	0,02	0,02	0,03	0,02	0,03	0,02	0,03	0,02
28	80	6	0,00	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02	0,02
28	80	7	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23	0,23
28	80	8	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	9	33,49	33,92	35,98	33,62	33,37	34,03	35,33	34,62	35,36	33,05	34,07	35,48	33,04	33,04	33,96	34,94
28	80	10	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	11	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	12	2,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	80	13	0,42	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00
28	79	1	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	2	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	3	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
28	79	4	0,00	2,48	0,43	2,77	3,03	2,37	1,07	1,78	1,04	3,35	2,33	0,91	3,36	3,36	2,44	1,46
		1-13	36,68	37,88	40,07	37,64	37,33	36,06	38,37	37,74	38,78	35,83	34,80	37,43	34,21	36,75	37,35	38,39
		5-4	36,68	36,68	36,69	36,68	36,68	36,68	36,69	36,68	36,68	36,68	36,68	36,68	36,68	36,68	36,68	36,68

Além dos tempos dos referidos pontos, existem outras duas linhas de somatórios: a primeira é o somatório dos tempos referentes a todos os pontos do período corrente, pontos de 1 a 13 (1-13); a segunda é o somatório entre dois eventos de sincronismo, que ocorrem no ponto 4. Logo, o somatório de um período completo entre dois pontos de sincronismo começa no ponto 5 de um período e termina no ponto 4 do período subseqüente da solução *backward* (5-4). Por conta de estar entre os dois pontos de sincronismo, os tempos referentes a esta linha são sempre praticamente iguais.

B.4 Comparação de Desempenho entre as Bibliotecas OSL e COIN

Neste item estão apresentado os resultados completos das convergências e dos comparativos entre as execuções com as versões 5 e 6 da estratégia de paralelização com as bibliotecas OSL, COIN 32 bits e COIN 64 bits para o caso de PMO de Março de 2009. Além deste caso, também estão mostrados os resultados para os casos de PMO de Abril e Agosto de 2010, porém sem a biblioteca COIN 32 bits.

- **Caso PMO de Março de 2009**

Os valores de cada iteração do processo de convergência para as três opções de bibliotecas do caso PMO de Março de 2009 estão mostrados na Tabela 54.

Tabela 54 – Convergências do Caso PMO Março de 2009 (Valores em 10^6 R$)

Iter.	Versão OSL				Versão COIN 32 bits				Versão COIN 64 bits			
	Limite Inferior	Zinf	Limite Superior	Zsup	Limite Inferior	Zinf	Limite Superior	Zsup	Limite Inferior	Zinf	Limite Superior	Zsup
1	70232,4	35804,4	99719,0	85629,6	70261,4	35804,4	99737,1	85657,9	70197,0	35810,4	99743,9	85661,5
2	59161,2	41347,4	85397,2	73149,6	59201,1	41313,4	85472,7	73214,2	59287,1	41349,2	85508,1	73275,1
3	58924,5	44168,8	82157,6	71281,6	59004,5	44112,4	82144,2	71287,1	59109,7	44105,2	82147,6	71300,0
4	53300,9	44408,2	75042,4	64866,4	53288,0	44348,3	75379,7	65104,4	53243,3	44346,3	74826,0	64684,2
5	53301,0	48546,9	75042,4	64866,4	53288,0	48390,8	75379,7	65104,4	53243,3	48527,4	74826,0	64684,2
6	52310,1	48581,6	73310,8	63424,8	52335,5	48413,7	73053,3	63273,7	52102,2	48551,3	72835,5	63097,3
7	52310,1	50646,3	73310,8	63424,8	52335,5	50971,5	73053,3	63273,7	52102,2	50775,5	72835,5	63097,3
8	51875,4	50675,8	72694,9	62935,1	51871,7	50991,8	72456,5	62794,7	51991,6	50812,9	72824,5	63071,0
9	51875,4	51150,6	72694,9	62935,0	51871,7	51206,9	72456,5	62794,7	51991,6	51142,8	72824,5	63071,0
10	51741,8	51171,5	72118,9	62565,7	51980,1	51231,8	72143,6	62607,2	52118,6	51199,1	72439,2	62885,8
11	51741,8	51471,2	72118,9	62565,7	51980,1	51511,9	72143,6	62607,2	52118,6	51428,3	72439,2	62885,8
12	51812,0	51488,2	71885,6	62446,3	51747,0	51541,4	72015,9	62504,9	51505,4	51455,4	71855,1	62389,3
13	51812,0	51655,7	71885,6	62446,3	51747,0	51705,0	72015,9	62504,9	51505,4	51649,4	71855,1	62389,3
14	51812,0	51711,9	71885,6	62446,3	51736,4	51731,6	71740,6	62333,2	-	-	-	-
15	51615,4	51772,1	71804,0	62421,8	51736,4	51915,6	71740,6	62333,2	-	-	-	-

Os resultados para este caso de PMO, para comparação do tempo de execução da última iteração comum, estão mostrados na Tabela 55, onde podem-se visualizar as 15 colunas, que significam o seguinte:

- Qte. Procs → quantidade de processadores utilizido na simulação.
- Parâmetros → define o tipo de resultado da referida linha: o parâmetro execução define o tempo total de processamento gasto no caso; o parâmetro Fator Acel. indica o fator de aceleração da simulação; e o parâmetro Eficiência indica as eficiências alcançadas nas simulações.
- OSL V5 – Val. → resultados obtidos com as simulações utilizando a versão 5 e a biblioteca OSL.
- COIN 32 bits V5 – Val. → resultados obtidos com as simulações utilizando a versão 5 e a biblioteca COIN de 32 bits.
- COIN 32 bits V5 – Rel. OSL V5 (%) → diferenças percentuais entre os tempos de execução das versões 5 com as bibliotecas COIN 32 bits e OSL. Resultados positivos indicam que a biblioteca COIN 32 bits levou mais tempo para resolver a iteração que a biblioteca OSL, enquanto que valores negativos significam o inverso.
- COIN 64 bits V5 – Val. → resultados obtidos com as simulações utilizando a versão 5 e a biblioteca COIN de 64 bits.
- COIN 64 bits V5 – Rel. OSL V5 (%) → diferenças percentuais entre os tempos de execução das versões 5 com as bibliotecas COIN 64 bits e OSL. Resultados positivos indicam que a biblioteca COIN 64 bits levou mais tempo para resolver a iteração que a biblioteca OSL, enquanto que valores negativos significam o inverso.

- OSL V6 – Val. → resultados obtidos com as simulações utilizando a versão 6 e a biblioteca OSL.
- OSL V6 – Rel. OSL V5 (%) → diferenças percentuais entre os tempos de execução das versões 5 e 6 com a biblioteca OSL. Resultados negativos indicam que a versão 6 levou menos tempo que a versão 5, enquanto que valores positivos significam o inverso.
- COIN 32 bits V6 – Val. → resultados obtidos com as simulações utilizando a versão 6 e a biblioteca COIN de 32 bits.
- COIN 32 bits V6 – Rel. OSL V5 (%) → diferenças percentuais entre os tempos de execução da versão 6 com a biblioteca COIN 32 bits e a versão 5 com a biblioteca OSL. Resultados positivos indicam que a biblioteca COIN 32 bits levou mais tempo para resolver a iteração que a biblioteca OSL, enquanto que valores negativos significam o inverso.
- COIN 32 bits V6 – Rel. OSL V6 (%) → diferenças percentuais entre os tempos de execução das versões 6 com as bibliotecas COIN 32 bits e OSL. Resultados positivos indicam que a biblioteca COIN 32 bits levou mais tempo para resolver a iteração que a biblioteca OSL, enquanto que valores negativos significam o inverso.
- COIN 64 bits V6 – Val. → resultados obtidos com as simulações utilizando a versão 6 e a biblioteca COIN de 64 bits.
- COIN 64 bits V6 – Rel. OSL V5 (%) → diferenças percentuais entre os tempos de execução da versão 6 com a biblioteca COIN 64 bits e a versão 5 com a biblioteca OSL. Resultados positivos indicam que a biblioteca COIN 64 bits levou mais tempo para resolver a iteração que a biblioteca OSL, enquanto que valores negativos significam o inverso.
- COIN 64 bits V6 – Rel. OSL V6 (%) → diferenças percentuais entre os tempos de execução das versões 6 com as bibliotecas COIN 64 bits e OSL. Resultados positivos indicam que a biblioteca COIN 64 bits levou mais tempo para resolver a iteração que a biblioteca OSL, enquanto que valores negativos significam o inverso.

As duas versões de 32 bits convergiram com 15 iterações, enquanto que a versão com 64 bits convergiu com 13 iterações. Por conta disto, a comparação dos tempos será feita com os valores obtidos para as 13as iterações de todos os casos, conforme pode ser visto na tabela a seguir. Apesar dos problemas resolvidos nesta iteração das diferentes

versões de bibliotecas não terem sido exatamente os mesmos, por conta das convergências diferentes, eles são bastante parecidos, logo, deveriam ser resolvidos, em média, em tempos similares. Este fato faz com que a análise dos tempos desta iteração seja válida para dar uma idéia dos desempenhos das versões com diferentes bibliotecas.

Tabela 55 – Valores Correspondentes à 13a. Iteração do Caso PMO de Março de 2009 para as Três Opções de Bibliotecas de Solução de PLs Utilizando as Versões 5 e 6

Qte. Procs.	Parâmetros	OSL V5	COIN 32 bits V5		COIN 64 bits V5		OSL V6		COIN 32 bits V6			COIN 64 bits V6		
		Val.	Val.	Rel OSL V5 (%)	Val	Rel OSL V5 (%)	Val.	Rel OSL V5 (%)	Val.	Rel OSL V5 (%)	Rel OSL V6 (%)	Val.	Rel OSL V5 (%)	Rel OSL V6 (%)
1	Execução (s)	8869	10050	13,32	7164	-19,22								
2	Execução (s)	4478	5136	14,69	3694	-17,51								
	Fator Acel.	1,98	1,96		1,94									
	Eficiência (%)	99,03	97,84		96,97									
4	Execução (s)	2263	2643	16,79	1909	-15,64								
	Fator Acel.	3,92	3,80		3,75									
	Eficiência (%)	97,98	95,06		93,82									
8	Execução (s)	1266	1452	14,69	1112	-12,16	1170	-7,58	1350	6,64	15,38	977	-22,83	-16,50
	Fator Acel.	7,01	6,92		6,44		7,58		7,44			7,33		
	Eficiência (%)	87,57	86,52		80,53		94,75		93,06			91,66		
16	Execução (s)	673	769	14,26	585	-13,08	624	-7,28	717	6,54	14,90	518	-23,03	-16,99
	Fator Acel.	13,18	13,07		12,25		14,21		14,02			13,83		
	Eficiência (%)	82,36	81,68		76,54		88,83		87,60			86,44		
32	Execução (s)	367	419	14,17	320	-12,81	343	-6,54	396	7,90	15,45	286	-22,07	-16,62
	Fator Acel.	24,17	23,99		22,39		25,86		25,38			25,05		
	Eficiência (%)	75,52	74,96		69,96		80,80		79,31			78,28		
40	Execução (s)	320	349	9,06	266	-16,88	286	-10,63	330	3,13	15,38	238	-25,63	-16,78
	Fator Acel.	27,72	28,80		26,93		31,01		30,45			30,10		
	Eficiência (%)	69,29	71,99		67,33		77,53		76,14			75,25		
64	Execução (s)	217	253	16,59	188	-13,36	204	-5,99	233	7,37	14,22	171	-21,20	-16,18
	Fator Acel.	40,87	39,72		38,11		43,48		43,13			41,89		
	Eficiência (%)	63,86	62,07		59,54		67,93		67,40			65,46		
80	Execução (s)	189	213	12,70	161	-14,81	179	-5,29	201	6,35	12,29	146	-22,75	-18,44
	Fator Acel.	46,93	47,18		44,50		49,55		50,00			49,07		
	Eficiência (%)	58,66	58,98		55,62		61,93		62,50			61,34		
128	Execução (s)	150	164	9,33	124	-17,33	144	-4,00	157	4,67	9,03	117	-22,00	-18,75
	Fator Acel.	59,13	61,28		57,77		61,59		64,01			61,23		
	Eficiência (%)	46,19	47,88		45,14		48,12		50,01			47,84		
200	Execução (s)	126	145	15,08	99	-21,43								
	Fator Acel.	70,39	69,31		72,36									
	Eficiência (%)	35,19	34,66		36,18									

- **Caso PMO de Abril de 2010**

Os valores completos de cada um dos dois processos de convergência do caso PMO de Abril de 2010 estão mostrados na Tabela 56.

Tabela 56 – Convergências do Caso PMO Abril de 2010 (Valores em 10^6 R$)

Iter.	Versão OSL				Versão COIN 64 bits			
	Limite Inferior	Zinf	Limite Superior	Zsup	Limite Inferior	Zinf	Limite Superior	Zsup
1	48752.27	26380.03	71698.40	61165.54	48889.37	26373.86	71843.82	61289.39
2	41408.84	30246.36	61467.27	52468.30	41663.22	30267.54	61773.97	52697.13
3	40314.68	31043.40	58446.53	50259.89	40406.19	30980.12	58130.76	50105.23
4	38735.23	31456.03	56162.69	48312.11	38516.98	31383.73	56153.39	48199.02
5	38735.23	32185.54	56162.69	48312.11	38516.98	32098.63	56153.39	48199.02
6	37999.11	32352.74	54846.40	47209.83	37777.27	32332.30	54800.27	47153.07
7	37999.11	34241.86	54846.40	47209.83	37777.27	34142.38	54800.27	47153.07
8	37215.96	34312.98	54264.25	46672.84	37412.04	34261.40	54103.31	46606.35
9	37215.96	35512.86	54264.25	46672.84	37412.04	35615.02	54103.31	46606.35
10	37037.25	35580.02	53616.60	46162.46	36874.42	35702.24	53438.91	46081.82
11	37037.25	35982.58	53616.60	46162.46	36874.42	36047.21	53438.91	46081.82
12	36974.48	36024.30	53246.60	45984.18	36796.79	36074.62	53117.22	45852.66
13	36974.48	36141.79	53246.60	45984.18	36796.79	36188.30	53117.22	45852.66
14	36652.14	36169.19	52878.36	45642.35	36684.70	36201.48	52870.67	45638.02
15	36652.14	36241.48	52878.36	45642.35	36684.70	36316.47	52870.67	45638.02
16	36652.14	36260.59	52878.36	45642.35	36684.70	36329.23	52870.67	45638.02
17	36652.14	36362.43	52878.36	45642.35	36684.70	36399.32	52870.67	45638.02
18	36735.37	36379.19	52761.93	45567.89	36684.70	36415.79	52870.67	45638.02
19	36735.37	36434.46	52761.93	45567.89	36684.70	36445.87	52870.67	45638.02
20	36602.70	36453.68	52568.36	45446.36	36605.54	36473.55	52497.91	45427.08
21	36602.70	36523.15	52568.36	45446.36	36605.54	36508.27	52497.91	45427.08
22	36602.70	36549.67	52568.36	45446.36	36605.54	36541.88	52497.91	45427.08
23	36602.70	36604.49	52568.36	45446.36	36605.54	36553.37	52497.91	45427.08
24	-	-	-	-	36605.54	36634.36	52497.91	45427.08

Os resultados da Tabela 57 mostram os tempos da última iteração comum deste caso de PMO com as bibliotecas OSL e COIN 64 bits. Nesta tabela podem se visualizar 10 colunas, cujos significados são similares aos das colunas da tabela anterior. As diferenças são as seguintes: não existem resultados para a biblioteca COIN 32 bits, logo, as colunas de resultados e diferenças percentuais relativas a esta biblioteca não existem; com relação à versão 6 com a biblioteca COIN 64 bits, existe uma comparação percentual com a versão 5 da mesma biblioteca e com a versão 6 da biblioteca OSL, em vez de uma comparação com as versões 5 e 6 da biblioteca OSL; não foram gerados resultados para as quantidades de 2, 4, 40 e 80 processadores.

As simulações com a biblioteca OSL convergiram com 23 iterações, enquanto que a convergência levou 24 iterações nos casos com a biblioteca COIN 64 bits. Por causa disto, a comparação dos tempos será feita com os valores obtidos nas 23as iterações de todos os casos.

Tabela 57 - Valores Correspondentes à 23a. Iteração do Caso PMO de Abril de 2010 para as Bibliotecas OSL e COIN 64 bits Utilizando as Versões 5 e 6

Qte. Procs.	Parâmetros	OSL V5	COIN 64 bits V5		OSL V6		COIN 64 bits V6		
		Valores	Valores	Relação OSL V5 (%)	Valores	Relação OSL V5 (%)	Valores	Relação COIN V5(%)	Relação OSL V6 (%)
1	Execução (s)	13448	12066	-10.28					
8	Execução (s)	2055	1972	-4.04	1788	-12.99	1640	-16.82	-8.26
	FA	6.54	6.12		7.52		7.36		
	Efic. (%)	81.80	76.49		94.02		91.95		
16	Execução (s)	1095	1045	-4.63	961	-12.26	876	-16.11	-8.81
	FA	12.28	11.55		13.99		13.77		
	Efic. (%)	76.74	72.19		87.46		86.06		
32	Execução (s)	596	576	-3.41	529	-11.35	487	-15.39	-7.82
	FA	22.55	20.95		25.44		24.76		
	Efic. (%)	70.47	65.46		79.49		77.37		
64	Execução (s)	355	338	-4.69	319	-10.23	292	-13.79	-8.47
	FA	37.88	35.66		42.20		41.37		
	Efic. (%)	59.19	55.73		65.94		64.64		
128	Execução (s)	245	226	-7.62	225	-8.16	202	-10.90	-10.37
	FA	54.89	53.31		59.77		59.83		
	Efic. (%)	42.88	41.65		46.70		46.74		
200	Execução (s)	218	185	-14.98					
	FA	61.69	65.11						
	Efic. (%)	30.84	32.55						

- **Caso PMO de Agosto de 2010**

Os valores completos de cada um dos dois processos de convergência do caso PMO de Agosto de 2010 estão mostrados na Tabela 58.

Tabela 58 – Convergências do Caso PMO Agosto de 2010 (Valores em 10^6 R$)

Iter.	Versão OSL				Versão COIN 64 bits			
	Limite Inferior	Zinf	Limite Superior	Zsup	Limite Inferior	Zinf	Limite Superior	Zsup
1	56901.43	28376.55	81458.57	70416.62	56919.19	28376.38	81502.75	70452.05
2	48039.36	33288.75	70027.95	60328.99	48070.15	33264.82	70217.83	60503.02
3	44872.40	34314.87	64148.63	55658.76	44948.85	34252.74	64316.65	55735.11
4	44015.05	34714.01	62997.46	54656.19	44006.89	34698.10	63289.26	54855.31
5	43513.17	35777.66	61931.71	53803.83	43363.81	35715.44	61980.26	53793.81
6	42707.72	35902.63	60911.86	52937.81	42815.42	35865.79	61318.34	53211.47
7	42707.72	37275.36	60911.86	52937.81	42603.61	37154.37	60834.07	52882.75
8	41997.58	37319.46	60178.42	52227.18	42225.68	37211.32	60199.92	52297.40
9	41997.58	38283.54	60178.42	52227.18	42092.21	38223.32	60016.55	52245.81
10	42070.41	38322.86	60098.25	52180.98	42001.82	38264.59	60132.49	52162.21
11	41570.56	38833.81	59587.50	51788.79	41573.85	38829.19	59171.46	51569.02
12	41367.43	38950.71	59153.17	51485.33	41573.85	38852.50	59171.46	51569.02
13	41238.63	39308.22	58696.71	51141.73	41573.85	39375.88	59171.46	51569.02
14	41238.63	39355.97	58696.71	51141.73	41291.41	39410.44	59068.44	51320.45
15	41238.63	39799.36	58696.71	51141.73	41065.36	39974.68	58927.39	51217.91
16	41238.63	39837.68	58696.71	51141.73	41065.36	40000.16	58927.39	51217.91
17	41238.63	40279.31	58696.71	51141.73	41247.00	40335.93	58653.94	51129.57
18	40958.11	40307.04	58506.89	50916.58	40787.33	40374.51	58332.79	50716.91
19	40857.94	40626.49	58306.14	50806.78	40787.33	40742.73	58332.79	50716.91
20	40857.94	40650.81	58306.14	50806.78	40787.33	40750.08	58332.79	50716.91
21	40810.39	40792.15	58244.41	50750.57	40502.96	40821.25	58250.91	50632.48
22	40506.32	40804.32	58082.82	50543.36	-	-	-	-

Os resultados da última iteração comum deste caso de PMO estão mostrados na Tabela 59, onde se podem visualizar 10 colunas, cujos significados são iguais aos das colunas da tabela anterior.

As simulações com a biblioteca OSL convergiram com 22 iterações, enquanto que nos casos com a biblioteca COIN 64 bits, a convergência levou 21 iterações. Por causa disto, a comparação dos tempos será feita com os valores obtidos nas 21as iterações de todos os casos.

Tabela 59 - Valores Correspondentes à 21ª Iteração do Caso PMO de Agosto de 2010 para as Bibliotecas OSL e COIN 64 bits Utilizando as Versões 5 e 6

Qte. Procs.	Parâmetros	OSL V5	COIN 64 bits V5		OSL V6		COIN 64 bits V6		
		Valores	Valores	Relação OSL V5 (%)	Valores	Relação OSL V5 (%)	Valores	Relação COIN V5(%)	Relação OSL V6 (%)
1	Execução (s)	12791	10739	-16.04					
8	Execução (s)	1906	1737	-8.88	1703	-10.67	1464	-15.74	-14.05
	FA	6.71	6.18		7.51		7.34		
	Efic. (%)	83.87	77.28		93.89		91.72		
16	Execução (s)	1015	920	-9.39	906	-10.80	780	-15.18	-13.84
	FA	12.60	11.67		14.12		13.76		
	Efic. (%)	78.74	72.96		88.27		86.02		
32	Execução (s)	553	503	-8.98	498	-9.95	434	-13.84	-12.92
	FA	23.13	21.34		25.69		24.76		
	Efic. (%)	72.28	66.68		80.27		77.39		
64	Execução (s)	329	296	-9.84	298	-9.23	257	-13.27	-13.85
	FA	38.92	36.24		42.88		41.79		
	Efic. (%)	60.81	56.63		66.99		65.29		
128	Execução (s)	225	195	-13.33	209	-7.11	174	-10.77	-16.75
	FA	56.85	55.07		61.20		61.72		
	Efic. (%)	44.41	43.03		47.81		48.22		
200	Execução (s)	193	158	-18.31					
	FA	66.28	68.11						
	Efic. (%)	33.14	34.06						

Anexo C

Detalhes dos Resultados das Simulações

Para a obtenção de resultados mais confiáveis, minimizando o risco de se obter valores de tempo muito discrepantes, foram feitas, pelo menos, três execuções para cada quantidade de processador de cada caso. De posse destes valores, foram calculados a média, o desvio padrão e relação entre estes dois valores. Caso a relação entre o desvio padrão e a média fosse maior que 0,5%, uma quarta simulação era feita e o valor mais discrepante era descartado, obtendo-se novos valores de média, desvio padrão e relação entre desvio padrão e média. Este procedimento foi adotado em todas as execuções no *cluster* do CEPEL, porém ao se executar os casos no servidor do NACAD, observaram-se valores muito diferentes nos tempos consumidos em determinadas iterações para diferentes execuções de um mesmo caso. Para que estes valores muito diferentes não gerassem distorções nas eficiências da estratégia de paralelização, o procedimento para obtenção dos tempos foi alterado. Em vez de se executar três vezes o caso, foram feitas apenas duas execuções, caso a relação entre o desvio padrão e a média fosse inferior a 0,5% nos tempos das iterações, os valores do caso que consumiu menos tempo seriam adotados. Caso fossem observadas relações muito elevadas entre o desvio padrão e a média, uma terceira simulação seria feita e o menor tempo de cada iteração seria adotado para compor o resultado final.

C.1 Versão 2 (CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	24	26	21	24	31	0	24	29	26				88.136	142	0,2
2	12	32	41	12	32	23	12	34	2				45.182	53	0,1
4	6	28	55	6	28	36	6	29	11				23.334	18	0,1
8	3	35	47	3	36	19	3	36	30				12.972	22	0,2
16	2	0	6	2	0	23	2	0	25				7.218	10	0,1
32	1	10	26	1	10	37	1	10	30				4.231	6	0,1
40	0	58	27	0	58	32	0	58	31				3.510	3	0,1
64	0	45	49	0	45	47	0	45	47				2.748	1	0,0
80	0	41	24	0	41	21	0	41	22				2.482	2	0,1
128	0	31	50	0	31	55	0	31	47				1.911	4	0,2
200	0	26	49	0	26	43	0	26	44				1.605	3	0,2

C.2 Versao 3 (CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	24	27	32	24	29	9	24	31	57				88.173	134	0,2
2	12	19	15	12	20	56	12	19	16				44.389	58	0,1
4	6	14	52	6	15	11	6	14	46				22.496	13	0,1
8	3	26	22	3	27	56	3	26	32				12.417	52	0,4
16	1	50	45	1	50	53	1	50	45				6.648	5	0,1
32	1	3	45	1	3	47	1	3	45				3.826	1	0,0
40	0	55	1	0	54	55	0	54	59				3.298	3	0,1
64	0	40	10	0	40	10	0	40	10				2.410	0	0,0
80	0	37	12	0	37	19	0	37	17				2.236	4	0,2
128	0	29	7	0	29	21	0	29	20				1.756	8	0,4
200	0	27	14	0	27	14	0	27	12				1.633	1	0,1

C.3 Versão 4 (CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	24	29	49	24	27	2	24	26	45				88.072	102	0,1
2	12	19	20	12	19	52	12	18	45				44.359	34	0,1
4	6	13	20	6	13	36	6	14	28				22.428	36	0,2
8	3	25	46	3	25	46	3	25	37				12.343	5	0,0
16	1	49	53	1	49	55	1	49	55				6.594	1	0,0
32	1	1	50	1	2	2	1	1	51				3.714	7	0,2
40	0	53	12	0	53	10	0	53	8				3.190	2	0,1
64	0	37	46	0	37	44	0	37	55				2.268	6	0,3
80	0	35	7	0	35	7	0	35	14				2.109	4	0,2
128	0	26	50	0	26	53	0	26	46				1.610	4	0,2
200	0	27	18	0	27	17	0	27	21				1.639	2	0,1

C.4 Versão 5 (CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	24	27	9	24	27	56	24	27	9				88.045	27	0,0
2	12	19	56	12	20	1	12	20	0				44.399	3	0,0
4	6	13	52	6	13	26	6	14	40				22.439	38	0,2
8	3	25	36	3	25	27	3	25	59				12.341	17	0,1
16	1	49	36	1	49	35	1	49	39				6.577	2	0,0
32	1	0	34	1	0	36	1	0	33				3.634	2	0,0
40	0	53	19	0	51	12	0	51	9	0	51	8	3.070	2	0,1
64	0	36	45	0	37	29	0	37	26	0	37	29	2.248	2	0,1
80	0	33	3	0	33	2	0	33	8				1.984	3	0,2
128	0	27	15	0	27	15	0	27	21				1.637	3	0,2
200	0	22	55	0	22	51	0	23	8	0	23	3	1.382	7	0,5

C.5 Versão 6 (CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	3	13	13	3	13	5	3	13	15				11.591	5	0,0
16	1	43	15	1	43	13	1	43	11				6.193	2	0,0
32	0	57	37	0	57	45	0	57	44				3.462	4	0,1
40	0	48	27	0	48	32	0	48	16				2.905	8	0,3
64	0	34	47	0	35	23	0	35	22	0	35	28	2.124	3	0,2
80	0	30	42	0	30	42	0	30	38				1.841	2	0,1
128	0	25	55	0	25	55	0	25	54				1.555	1	0,0

C.6 Caso PMO Março de 2009 (COIN 32 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	27	7	14	27	6	28	27	4	18				97.560	91	0,1
2	13	51	6	13	52	47	13	53	43				49.952	80	0,2
4	7	6	51	7	3	36	7	3	38				25.482	112	0,4
8	3	52	19	3	52	23	3	52	35				13.946	8	0,1
16	2	3	17	2	3	17	2	3	14				7.396	2	0,0
32	1	7	53	1	7	58	1	7	53				4.075	3	0,1
40	0	57	7	0	57	7	0	57	7				3.427	0	0,0
64	0	42	29	0	41	35	0	41	46	0	41	44	2.502	6	0,2
80	0	36	0	0	35	46	0	35	47				2.151	8	0,4
128	0	29	8	0	29	2	0	29	5				1.745	3	0,2
200	0	25	50	0	26	4	0	26	22	0	26	22	1.576	10	0,7

C.7 Caso PMO Março de 2009 (COIN 32 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	3	38	17	3	38	26	3	38	40				13.108	12	0,1
16	1	56	8	1	56	3	1	56	15				6.969	6	0,1
32	1	4	37	1	4	36	1	4	41				3.878	3	0,1
40	0	54	22	0	54	10	0	54	27				3.260	9	0,3
64	0	39	23	0	39	21	0	39	21				2.362	1	0,0
80	0	33	40	0	34	53	0	34	45	0	34	45	2.088	5	0,2
128	0	27	35	0	27	28	0	27	36				1.653	4	0,3

C.8 Caso PMO Março de 2009 (COIN 64 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	14	50	48	14	53	29	14	51	9				53.509	88	0,2
2	7	39	15	7	39	26	7	38	29				27.543	30	0,1
4	3	54	30	3	54	33	3	54	45				14.076	8	0,1
8	2	13	45	2	13	29	2	13	47				8.020	10	0,1
16	1	10	35	1	10	43	1	10	43				4.240	5	0,1
32	0	39	25	0	39	20	0	39	20				2.362	3	0,1
40	0	33	26	0	33	23	0	33	23				2.004	2	0,1
64	0	25	2	0	24	52	0	24	54				1.496	5	0,4
80	0	21	25	0	21	26	0	21	33				1.288	4	0,3
128	0	18	20	0	18	27	0	18	11	0	18	16	1.096	5	0,4
200	0	14	52	0	15	12	0	15	5	0	15	5	907	4	0,4

C.9 Caso PMO Março de 2009 (COIN 64 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	2	0	46	2	0	42	2	0	49				7.246	4	0,0
16	1	4	6	1	4	6	1	4	4				3.845	1	0,0
32	0	36	11	0	36	13	0	36	10				2.171	2	0,1
40	0	30	29	0	30	28	0	30	30				1.829	1	0,1
64	0	22	49	0	22	47	0	22	46				1.367	2	0,1
80	0	19	25	0	19	35	0	19	23	0	19	23	1.164	1	0,1
128	0	16	59	0	16	54	0	16	58				1.017	3	0,3

C.10 Caso PMO Março de 2009 (300 Séries - OSL – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	60	32	48	60	53	58	60	49	6				218.717	665	0,3
8	8	40	37	8	42	4	8	43	12				31.318	78	0,2
16	4	32	39	4	32	46	4	32	41				16.362	4	0,0
32	2	25	23	2	25	34	2	25	26				8.728	6	0,1
64	1	22	58	1	23	48	1	23	39	1	23	48	5.025	5	0,1
128	0	54	38	0	54	32	0	54	20				3.270	9	0,3
200	0	44	58	0	44	45	0	44	46				2.690	7	0,3

C.11 Caso PMO Março de 2009 (300 Séries - OSL – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	7	53	8	7	54	24	7	56	10				28.474	91	0,3
16	4	8	58	4	8	59	4	9	19				14.945	12	0,1
32	2	13	42	2	13	44	2	13	42				8.023	1	0,0
64	1	16	24	1	17	9	1	17	5	1	17	12	4.629	4	0,1
128	0	50	22	0	50	19	0	50	17				3.019	3	0,1

C.12 Caso PMO Março de 2009 (300 Séries - COIN 64 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	42	15	59	42	16	38	42	16	40				152.186	23	0,0
8	6	32	1	6	32	2	6	32	8				23.524	4	0,0
16	3	23	50	3	24	31	3	24	22				12.254	22	0,2
32	1	49	26	1	49	35	1	49	32				6.571	5	0,1
64	1	2	48	1	2	54	1	2	44				3.769	5	0,1
128	0	41	5	0	41	7	0	40	57				2.463	5	0,2
200	0	32	56	0	32	49	0	32	55				1.973	4	0,2

C.13 Caso PMO Março de 2009 (300 Séries – COIN 64 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	5	39	43	5	39	33	5	39	44				20.380	6	0,0
16	2	58	11	2	58	10	2	58	9				10.690	1	0,0
32	1	36	45	1	36	48	1	36	49				5.807	2	0,0
64	0	55	37	0	55	36	0	55	42				3.338	3	0,1
128	0	36	47	0	36	43	0	36	42				2.204	3	0,1

C.14 Caso PMO Março de 2009 (50 Aberturas - OSL – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	189	42	24	190	27	5	190	12	41				684.443	1368	0,2
8	27	25	8	27	25	19	27	27	56				98.768	94	0,1
16	14	18	39	14	18	54	14	17	40				51.504	39	0,1
32	7	45	25	7	46	12	7	45	37				27.945	24	0,1
64	4	24	0	4	24	35	4	24	22				15.859	18	0,1
128	2	52	6	2	52	19	2	52	3				10.329	9	0,1
200	2	6	53	2	6	43	2	6	42				7.606	6	0,1

C.15 Caso PMO Março de 2009 (50 Aberturas - OSL – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	24	34	55	24	35	36	24	35	46				88.526	27	0,0
16	12	52	46	12	52	41	12	52	31				46.359	8	0,0
32	7	4	17	7	4	7	7	4	22				25.455	8	0,0
64	4	2	18	4	3	4	4	3	5				14.569	27	0,2
128	2	41	19	2	41	22	2	41	24				9.682	3	0,0

C.16 Caso PMO Março de 2009 (50 Aberturas – COIN 64 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	167	0	29	166	36	9	166	49	41				600.526	732	0,1
8	26	19	41	26	18	14	26	20	41				94.772	74	0,1
16	13	37	22	13	38	4	13	37	51				49.066	22	0,0
32	7	20	32	7	21	17	7	20	38				26.449	24	0,1
64	4	8	25	4	8	27	4	8	41				14.911	9	0,1
128	2	35	47	2	36	1	2	35	53				9.354	7	0,1
200	1	53	32	1	53	24	1	53	36				6.811	6	0,1

C.17 Caso PMO Março de 2009 (50 Aberturas – COIN 64 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	22	14	18	22	16	2	22	14	16				80.092	61	0,1
16	11	35	20	11	34	50	11	35	11				41.707	15	0,0
32	6	21	53	6	21	24	6	21	9				22.889	22	0,1
64	3	37	51	3	37	53	3	37	58				13.074	4	0,0
128	2	20	22	2	20	8	2	20	21				8.417	8	0,1

C.18 Caso PMO Março de 2009 (300 Séries e 50 Aberturas - OSL – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	1117	38	0										4.023.480	-	-
8	227	34	49										819.289	-	-
16	116	13	25										418.405	-	-
32	59	21	26										213.686	-	-
64	31	58	36										115.116	-	-
128	18	26	24										66.384	-	-
200	13	50	2										49.802	-	-

C.19 Caso PMO Março de 2009 (300 Séries e 50 Aberturas - OSL – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	144	1	30										518.490	-	-
16	75	55	30										273.330	-	-
32	41	5	30										147.930	-	-
64	22	8	16										79.696	-	-
128	13	36	35										48.995	-	-

C.20 Caso PMO Março de 2009 (300 Séries e 50 Aberturas – COIN 64 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	925	29	10										3.331.750	-	-
8	213	21	48										768.108	-	-
16	109	22	51										393.771	-	-
32	54	38	13										196.693	-	-
64	29	19	54										105.594	-	-
128	16	50	36										60.636	-	-
200	12	21	14										44.474	-	-

C.21 Caso PMO Março de 2009 (300 Séries e 50 Aberturas – COIN 64 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	122	25	39										440.739	-	-
16	63	44	0										229.440	-	-
32	33	56	14										122.174	-	-
64	18	41	28										67.288	-	-
128	11	28	19										41.299	-	-

C.22 Caso PMO Abril de 2010 (OSL – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	47	23	54	47	27	3	47	27	0				170.759	108	0,1
8	6	54	58	6	54	35	6	55	47				24.907	37	0,1
16	3	41	46	3	42	1	3	42	21				13.323	18	0,1
32	2	1	24	2	1	25	2	1	43				7.291	11	0,1
64	1	14	53	1	14	28	1	14	27				4.476	15	0,3
128	0	53	13	0	53	15	0	53	15				3.194	1	0,0
200	0	45	57	0	45	42	0	48	54	0	45	47	2.749	8	0,3

C.23 Caso PMO Abril de 2010 (OSL – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	6	16	34	6	16	50	6	16	36				22.600	9	0,0
16	3	22	23	3	22	1	3	22	14				12.133	11	0,1
32	1	52	10	1	52	13	1	52	13				6.732	2	0,0
64	1	8	25	1	8	21	1	8	14				4.100	8	0,2
128	0	49	46	0	49	39	0	49	35				2.980	6	0,2

C.24 Caso PMO Abril de 2010 (COIN 64 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	44	23	24	44	17	53	44	15	58				159.545	232	0,1
8	7	0	8	7	0	39	6	58	31				25.186	67	0,3
16	3	42	15	3	41	43	3	42	16				13.325	19	0,1
32	2	3	30	2	3	35	2	2	10	2	3	2	7.402	18	0,2
64	1	14	13	1	14	21	1	13	52				4.449	15	0,3
128	0	51	43	0	51	31	0	51	41				3.098	6	0,2
200	0	42	1	0	42	7	0	42	13				2.527	6	0,2

C.25 Caso PMO Abril de 2010 (COIN 64 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	6	1	52	6	1	33	6	1	37				21.701	10	0,0
16	3	12	21	3	12	52	3	12	52				11.562	18	0,2
32	1	47	46	1	47	48	1	47	45				6.466	2	0,0
64	1	5	25	1	5	26	1	5	13				3.921	7	0,2
128	0	46	45	0	46	39	0	46	36				2.798	2	0,1

C.26 Caso PMO Agosto de 2010 (OSL – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	44	47	57	44	48	29	44	50	55				161.347	95	0,1
8	6	28	26	6	28	22	6	29	17				23.322	31	0,1
16	3	27	26	3	27	6	3	27	2				12.431	13	0,1
32	1	54	13	1	53	43	1	54	6				6.841	16	0,2
64	1	9	16	1	9	14	1	9	33				4.161	10	0,3
128	0	49	11	0	49	10	0	49	17				2.953	4	0,1
200	0	42	22	0	42	1	0	41	47	0	42	10	2.531	11	0,4

C.27 Caso PMO Agosto de 2010 (OSL – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	5	55	37	5	54	46	5	54	56				21.306	27	0,1
16	3	10	2	3	10	8	3	10	30				11.413	15	0,1
32	1	45	38	1	45	32	1	45	50				6.340	9	0,1
64	1	4	14	1	4	20	1	4	9				3.852	4	0,1
128	0	46	2	0	45	55	0	46	0				2.759	4	0,1

C.28 Caso PMO Agosto de 2010 (COIN 64 bits – Versão 5 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
1	33	38	40	33	39	43	33	39	47				121.163	38	0,0
8	5	14	55	5	14	1	5	15	5				18.880	34	0,2
16	2	46	14	2	46	23	2	46	26				9.980	6	0,1
32	1	31	46	1	31	40	1	32	22				5.516	23	0,4
64	0	55	53	0	55	50	0	55	31				3.345	12	0,4
128	0	38	42	0	38	34	0	38	43				2.320	5	0,2
200	0	31	24	0	31	25	0	31	21				1.883	2	0,1

C.29 Caso PMO Agosto de 2010 (COIN 64 bits – Versão 6 – CEPEL)

Qte. Procs.	Execução 1			Execução 2			Execução 3			Execução 4			Tempo Médio (s)	Desvio Padrão (s)	Relação Desvio/ Média (%)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg			
8	4	34	39	4	34	38	4	34	32				16.476	4	0,0
16	2	25	55	2	26	8	2	26	9				8.764	8	0,1
32	1	21	56	1	21	53	1	21	46				4.912	5	0,1
64	0	49	29	0	49	38	0	49	39				2.975	6	0,2
128	0	34	54	0	35	0	0	35	2				2.101	1	0,1

C.30 Caso PMO Março de 2009 (OSL – Versão 5 – NACAD – 1 Processador)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	7	0	0	8	0	0	7	-	-	-	7
1	0	18	55	0	18	43	0	18	57	1.129	8,49	0,75	1.123
2	0	26	40	0	26	33	0	26	43	1.597	4,95	0,31	1.593
3	0	33	27	0	33	27	0	34	42	2.007	0,00	0,00	2.007
4	0	44	42	0	44	50	0	44	53	2.686	5,66	0,21	2.682
5	0	50	6	0	50	1	0	55	24	3.004	3,54	0,12	3.001
6	1	3	33	1	8	46	1	3	51	3.970	221,32	5,58	3.813
7	1	9	34	1	9	31	1	11	8	4.173	2,12	0,05	4.171
8	1	18	48	1	19	1	1	19	0	4.735	9,19	0,19	4.728
9	1	26	55	1	36	0	1	26	55	5.488	385,37	7,02	5.215
10	1	35	21	1	45	12	1	45	15	6.017	417,90	6,95	5.721
11	1	44	51	1	45	47	1	49	57	6.319	39,60	0,63	6.291
12	1	52	44	1	57	23	1	52	53	6.904	197,28	2,86	6.764
13	2	2	44	2	11	3	2	2	30	7.614	352,85	4,63	7.350
14	2	9	44	2	9	43	2	9	53	7.784	0,71	0,01	7.783
15	2	21	25	2	23	47	2	21	7	8.556	100,41	1,17	8.467
SF	0	41	42	0	42	26	0	41	43	2.524	31,11	1,23	2.502
Total													73.218

C.31 Caso PMO Março de 2009 (OSL – Versão 5 – NACAD – 8 Processadores)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Tempos da Execução Mais Rápida (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	8	0	0	8				-	-	-	8
1	0	2	28	0	2	28				148	0,00	0,00	148
2	0	3	34	0	3	34				214	0,00	0,00	214
3	0	4	30	0	4	30				270	0,00	0,00	270
4	0	6	2	0	6	1				362	0,71	0,20	361
5	0	6	45	0	6	45				405	0,00	0,00	405
6	0	8	37	0	8	38				518	0,71	0,14	518
7	0	9	28	0	9	28				568	0,00	0,00	568
8	0	10	46	0	10	46				646	0,00	0,00	646
9	0	11	54	0	11	55				715	0,71	0,10	715
10	0	13	5	0	13	5				785	0,00	0,00	785
11	0	14	24	0	14	23				864	0,71	0,08	863
12	0	15	35	0	15	34				935	0,71	0,08	934
13	0	16	54	0	16	53				1.014	0,71	0,07	1.013
14	0	17	56	0	17	56				1.076	0,00	0,00	1.076
15	0	19	30	0	19	31				1.171	0,71	0,06	1.171
SF	0	5	43	0	5	43				343	0,00	0,00	343
Total													10.038

C.32 Caso PMO Março de 2009 (COIN 64 bits – Versão 5 – NACAD – 1 Processador)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Tempos da Execução Mais Rápida (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	4	0	0	4				-	-	-	4
1	0	11	17	0	11	18				677	0,00	0,00	677
2	0	18	3	0	18	6				1.079	6,36	0,59	1.083
3	0	24	11	0	24	11				1.447	6,36	0,44	1.451
4	0	31	9	0	31	14				1.865	6,36	0,34	1.869
5	0	37	27	0	37	19				2.240	9,90	0,44	2.247
6	0	45	53	0	45	53				2.754	0,71	0,03	2.753
7	0	50	49	0	50	41				3.047	3,54	0,12	3.049
8	1	0	0	1	0	2				3.600	0,00	0,00	3.600
9	1	5	23	1	5	23				3.885	54,45	1,40	3.923
10	1	12	11	1	12	14				4.284	66,47	1,55	4.331
11	1	18	21	1	18	42				4.651	71,42	1,54	4.701
12	1	26	26	1	26	30				5.140	65,76	1,28	5.186
13	1	31	4	1	31	10				5.426	54,45	1,00	5.464
SF	0	24	1	0	24	0				1.458	24,04	1,65	1.441
Total													41.775

C.33 Caso PMO Março de 2009 (COIN 64 bits – Versão 5 – NACAD – 8 Processadores)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	5	0	0	5	0	0	5	-	-	-	5
1	0	1	29	0	1	30	0	1	29	90	0,71	0,79	89
2	0	2	25	0	2	24	0	2	25	145	0,71	0,49	144
3	0	3	14	0	3	15	0	3	14	195	0,71	0,36	194
4	0	4	11	0	4	12	0	4	13	252	0,71	0,28	251
5	0	5	4	0	5	4	0	5	4	304	0,00	0,00	304
6	0	6	13	0	6	14	0	6	15	374	0,71	0,19	373
7	0	6	54	0	6	56	0	6	55	415	1,41	0,34	414
8	0	8	11	0	8	12	0	8	12	492	0,71	0,14	491
9	0	8	47	0	8	58	0	8	59	533	7,78	1,46	527
10	0	9	45	0	9	55	0	9	55	590	7,07	1,20	585
11	0	10	34	0	10	47	0	10	48	641	9,19	1,44	634
12	0	11	43	0	11	55	0	11	55	709	8,49	1,20	703
13	0	12	23	0	12	37	0	12	36	750	9,90	1,32	743
SF	0	3	26	0	3	20	0	3	21	203	4,24	2,09	200
Total													5.657

C.34 Caso PMO Março de 2009 (300 Séries - OSL – Versão 5 – NACAD – 1 Processador)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	9	0	0	9	0	0	8	-	-	-	8
1	0	37	8	0	35	2	0	35	9	2.165	89,10	4,12	2.102
2	0	51	59	0	51	46	0	51	45	3.113	9,19	0,30	3.105
3	1	9	35	1	9	43	1	9	50	4.179	5,66	0,14	4.175
4	1	35	21	1	35	26	1	35	10	5.724	3,54	0,06	5.710
5	1	48	2	1	52	36	1	49	17	6.619	193,75	2,93	6.482
6	2	18	51	2	25	23	2	18	41	8.527	277,19	3,25	8.321
7	2	30	54	2	30	57	2	30	33	9.056	2,12	0,02	9.033
8	2	53	56	3	9	0	2	53	7	10.888	639,22	5,87	10.387
9	3	10	18	3	10	56	3	10	13	11.437	26,87	0,23	11.413
10	3	52	29	3	34	10	3	33	28	13.400	777,11	5,80	12.808
11	3	51	36	4	17	15	3	51	39	14.666	1088,24	7,42	13.896
12	4	24	50	4	13	25	4	12	29	15.548	484,37	3,12	15.149
13	4	36	7	4	43	30	4	36	58	16.789	313,25	1,87	16.567
14	5	25	15	4	56	36	4	57	34	18.656	1215,52	6,52	17.796
15	5	17	12	5	15	41	5	47	48	18.987	64,35	0,34	18.941
16	5	34	52	6	9	36	5	39	6	21.134	1473,61	6,97	20.092
SF	1	3	38	1	3	41	1	3	51	3.820	2,12	0,06	3.818
Total													177.701

C.35 Caso PMO Março de 2009 (300 Séries - OSL – Versão 5 – NACAD – 8 Processadores)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Tempos da Execução Mais Rápida (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	8	0	0	8							8
1	0	4	34	0	4	34				274	0,00	0,00	274
2	0	6	51	0	6	51				411	0,00	0,00	411
3	0	9	17	0	9	17				557	0,00	0,00	557
4	0	12	44	0	12	45				765	0,71	0,09	764
5	0	14	35	0	14	37				876	1,41	0,16	875
6	0	18	44	0	18	45				1.125	0,71	0,06	1.124
7	0	20	23	0	20	25				1.224	1,41	0,12	1.223
8	0	23	32	0	23	31				1.412	0,71	0,05	1.412
9	0	25	59	0	25	59				1.559	0,00	0,00	1.559
10	0	29	9	0	29	10				1.750	0,71	0,04	1.749
11	0	31	42	0	31	43				1.903	0,71	0,04	1.902
12	0	34	42	0	34	40				2.081	1,41	0,07	2.082
13	0	37	51	0	37	52				2.272	0,71	0,03	2.271
14	0	40	23	0	40	22				2.423	0,71	0,03	2.423
15	0	43	16	0	43	17				2.597	0,71	0,03	2.596
16	0	46	11	0	46	5				2.768	4,24	0,15	2.771
SF	0	8	44	0	8	44				524	0,00	0,00	524
Total													24.525

C.36 Caso PMO Março de 2009 (300 Séries - COIN 64 bits – Versão 5 – NACAD – 1 Processador)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	6	0	0	5	0	0	5	-	-	-	5
1	0	28	28	0	28	28	0	25	27	1.708	0,00	0,00	1.527
2	0	40	40	0	36	43	0	36	36	2.322	167,58	7,22	2.196
3	0	55	51	0	54	9	0	50	4	3.300	72,12	2,19	3.004
4	1	12	49	1	13	0	1	5	21	4.375	7,78	0,18	3.921
5	1	27	32	1	27	30	1	18	58	5.251	1,41	0,03	4.738
6	1	47	32	1	47	15	1	36	42	6.444	12,02	0,19	5.802
7	1	59	43	1	59	52	1	47	45	7.188	6,36	0,09	6.465
8	2	21	22	2	21	14	2	6	56	8.478	5,66	0,07	7.616
9	2	34	31	2	34	27	2	18	47	9.269	2,83	0,03	8.327
10	2	51	21	2	51	26	2	34	5	10.284	3,54	0,03	9.245
11	3	7	32	2	55	17	2	48	21	10.885	519,72	4,77	10.101
12	3	24	45	3	4	19	3	4	6	11.672	866,91	7,43	11.046
13	3	41	0	3	19	1	3	19	13	12.601	932,67	7,40	11.941
14	3	59	30	3	35	46	3	35	52	13.658	1006,92	7,37	12.946
15	4	15	21	3	49	3	3	49	48	14.532	1115,81	7,68	13.743
SF	0	39	16	0	38	36	0	38	51	2.336	28,28	1,21	2.316
Total													114.939

C.37 Caso PMO Março de 2009 (300 Séries – COIN 64 bits – Versão 5 – NACAD – 8 Processadores)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	5	0	0	7	0	0	5	-	-	-	5
1	0	3	21	0	3	40	0	3	20	211	13,44	6,38	200
2	0	4	52	0	5	1	0	4	51	297	6,36	2,15	291
3	0	6	41	0	6	50	0	6	41	406	6,36	1,57	401
4	0	8	48	0	8	58	0	8	48	533	7,07	1,33	528
5	0	10	39	0	10	46	0	10	36	643	4,95	0,77	636
6	0	13	4	0	13	12	0	13	4	788	5,66	0,72	784
7	0	14	40	0	14	46	0	14	39	883	4,24	0,48	879
8	0	17	18	0	17	24	0	17	18	1.041	4,24	0,41	1.038
9	0	18	59	0	19	5	0	18	57	1.142	4,24	0,37	1.137
10	0	21	7	0	21	6	0	21	7	1.267	0,71	0,06	1.266
11	0	23	8	0	23	6	0	23	6	1.387	1,41	0,10	1.386
12	0	25	19	0	25	15	0	25	16	1.517	2,83	0,19	1.515
13	0	27	20	0	27	20	0	27	19	1.640	0,00	0,00	1.639
14	0	29	41	0	29	40	0	29	38	1.781	0,71	0,04	1.778
15	0	31	40	0	31	40	0	31	41	1.900	0,00	0,00	1.900
SF	0	5	27	0	5	27	0	5	26	327	0,00	0,00	326
Total													15.709

C.38 Caso PMO Março de 2009 (50 Aberturas - OSL – Versão 5 – NACAD – 1 Processador)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic.	0	0	8	0	0	8	0	0	8	-	-	-	8
1	0	44	33	0	44	47	0	44	45	2.680	9,90	0,37	2.673
2	1	4	5	1	3	56	1	3	47	3.841	6,36	0,17	3.827
3	1	23	25	1	22	24	1	23	22	4.975	43,13	0,87	4.944
4	1	46	17	1	46	18	1	45	13	6.378	0,71	0,01	6.313
5	2	5	15	2	18	59	2	5	0	7.927	582,66	7,35	7.500
6	2	30	59	2	32	52	2	28	40	9.116	79,90	0,88	8.920
7	2	49	3	3	7	25	2	47	52	10.694	779,23	7,29	10.072
8	3	12	1	3	31	40	3	8	34	12.111	833,68	6,88	11.314
9	3	33	6	3	33	29	3	31	42	12.798	16,26	0,13	12.702
10	3	50	29	3	50	20	3	50	49	13.825	6,36	0,05	13.820
11	4	16	32	4	15	59	4	44	17	15.376	23,33	0,15	15.359
12	4	31	25	4	31	13	4	32	4	16.279	8,49	0,05	16.273
13	5	1	17	5	34	57	5	7	58	19.087	1428,36	7,48	18.077
14	5	14	38	5	32	40	5	49	18	19.419	765,09	3,94	18.878
15	6	14	37	5	46	45	5	46	9	21.641	1182,28	5,46	20.769
16	6	10	15	6	32	0	6	10	23	22.868	922,77	4,04	22.215
17	6	36	1	6	38	9	6	31	30	23.825	90,51	0,38	23.490
18	7	16	46	6	45	29	6	41	2	25.268	1327,24	5,25	24.062
19	7	7	14	7	58	39	7	11	1	27.177	2181,42	8,03	25.634
20	7	25	59	7	26	13	8	16	10	26.766	9,90	0,04	26.759
21	8	43	36	7	55	1	8	50	9	29.959	2061,22	6,88	28.501
22	8	25	23	8	11	20	8	10	40	29.902	596,09	1,99	29.440
23	8	39	46	8	35	43	8	38	38	31.065	171,83	0,55	30.943
24	9	54	28	8	54	19	8	56	59	33.864	2551,95	7,54	32.059
25	9	25	3	9	52	53	9	21	51	34.738	1180,87	3,40	33.711
26	10	5	7	9	48	14	9	41	21	35.801	716,30	2,00	34.881
27	11	15	39	10	12	15	10	52	50	38.637	2689,83	6,96	36.735
28	11	24	28	10	28	20	10	57	9	39.384	2381,54	6,05	37.700
SF	1	14	38	1	20	50	1	23	7	4.664	263,04	5,64	4.478
Total													562.057

C.39 Caso PMO Março de 2009 (50 Aberturas - OSL – Versão 5 – NACAD – 8 Processadores)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Tempos da Execução Mais Rápida (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic	0	0	8	0	0	8				-	-	-	8
1	0	5	48	0	5	49				349	0,71	0,20	348
2	0	8	28	0	8	28				508	0,00	0,00	508
3	0	11	2	0	11	1				662	0,71	0,11	662
4	0	14	5	0	14	4				845	0,71	0,08	845
5	0	16	41	0	16	42				1.002	0,71	0,07	1.001
6	0	20	8	0	20	8				1.208	0,00	0,00	1.208
7	0	22	41	0	22	40				1.361	0,71	0,05	1.361
8	0	25	48	0	25	44				1.546	2,83	0,18	1.548
9	0	28	44	0	28	46				1.725	1,41	0,08	1.724
10	0	31	7	0	31	12				1.870	3,54	0,19	1.867
11	0	34	38	0	34	39				2.079	0,71	0,03	2.078
12	0	36	47	0	36	48				2.208	0,71	0,03	2.207
13	0	40	48	0	40	49				2.449	0,71	0,03	2.448
14	0	42	52	0	42	49				2.571	2,12	0,08	2.572
15	0	47	2	0	47	6				2.824	2,83	0,10	2.822
16	0	48	25	0	48	22				2.904	2,12	0,07	2.905
17	0	53	23	0	53	22				3.203	0,71	0,02	3.203
18	0	54	42	0	54	41				3.282	0,71	0,02	3.282
19	0	58	42	0	58	41				3.522	0,71	0,02	3.522
20	1	1	10	1	1	9				3.670	0,71	0,02	3.670
21	1	5	13	1	5	20				3.917	4,95	0,13	3.913
22	1	7	40	1	7	45				4.063	3,54	0,09	4.060
23	1	11	16	1	11	23				4.280	4,95	0,12	4.276
24	1	13	43	1	13	55				4.429	8,49	0,19	4.423
25	1	17	41	1	17	38				4.660	2,12	0,05	4.661
26	1	19	50	1	19	48				4.789	1,41	0,03	4.790
27	1	24	37	1	24	38				5.078	0,71	0,01	5.077
28	1	26	14	1	26	18				5.176	2,83	0,05	5.174
SF	0	10	18	0	10	17				618	0,71	0,11	618
Total													76.781

C.40 Caso PMO Março de 2009 (50 Aberturas – COIN 64 bits – Versão 5 – NACAD – 1 Processador)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Mínimo das 3 Execuções (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic	0	0	5	0	0	7	0	0	5	-	-	-	5
1	0	27	14	0	27	14	0	27	27	1.634	0,00	0,00	1.627
2	0	43	30	0	43	35	0	43	27	2.613	3,54	0,14	2.607
3	0	58	52	0	59	27	0	59	15	3.550	24,75	0,70	3.532
4	1	14	56	1	15	36	1	15	20	4.516	28,28	0,63	4.496
5	1	31	18	1	31	27	1	31	34	5.483	6,36	0,12	5.478
6	1	50	2	1	50	38	1	50	14	6.620	25,46	0,38	6.602
7	2	2	44	2	2	49	2	2	46	7.367	3,54	0,05	7.364
8	2	24	58	2	27	38	2	25	38	8.778	113,14	1,29	8.698
9	2	37	17	2	37	25	2	37	13	9.441	5,66	0,06	9.433
10	2	55	13	2	55	6	2	55	13	10.510	4,95	0,05	10.506
11	3	10	38	3	10	37	3	10	9	11.438	0,71	0,01	11.409
12	3	28	21	3	27	59	3	28	9	12.490	15,56	0,12	12.479
13	3	42	19	3	42	17	3	42	11	13.338	1,41	0,01	13.331
14	4	1	39	4	1	42	4	1	31	14.501	2,12	0,01	14.491
15	4	13	53	4	14	17	4	14	23	15.245	16,97	0,11	15.233
16	4	30	9	4	32	26	4	32	22	16.278	96,87	0,60	16.209
17	4	44	0	4	47	16	4	47	53	17.138	138,59	0,81	17.040
18	5	4	43	5	4	4	5	4	33	18.264	27,58	0,15	18.244
19	5	20	36	5	20	27	5	20	24	19.232	6,36	0,03	19.224
20	5	38	3	5	40	4	5	37	31	20.344	85,56	0,42	20.251
21	5	51	26	5	53	4	5	53	7	21.135	69,30	0,33	21.086
22	6	8	29	6	10	50	6	10	26	22.180	99,70	0,45	22.109
23	6	26	41	6	27	47	6	27	15	23.234	46,67	0,20	23.201
24	6	45	21	6	45	42	6	43	12	24.332	14,85	0,06	24.192
25	7	1	46	7	2	53	6	59	56	25.340	47,38	0,19	25.196
26	7	20	21	8	2	20	7	19	48	27.681	1781,20	6,43	26.388
27	7	35	36	8	27	36	7	33	11	28.896	2206,17	7,63	27.191
28	8	11	11	8	33	25	7	51	38	30.138	943,28	3,13	28.298
29	8	10	25	8	10	58	8	8	4	29.442	23,33	0,08	29.284
SF	0	54	36	0	54	10	0	54	20	3.263	18,38	0,56	3.250
Total													448.454

C.41 Caso PMO Março de 2009 (50 Aberturas – COIN 64 bits – Versão 5 – NACAD – 8 Processadores)

Iter.	Execução 1			Execução 2			Execução 3			Média 1-2 (s)	Desvio Padrão 1-2 (s)	Relação Desvio/ Média (%)	Tempos da Execução Mais Rápida (s)
	Hrs	Min	Seg	Hrs	Min	Seg	Hrs	Min	Seg				
Inic	0	0	5	0	0	7				-	-	-	5
1	0	3	32	0	3	32				212	0,00	0,00	212
2	0	5	44	0	5	44				344	0,00	0,00	344
3	0	7	47	0	7	47				467	0,00	0,00	467
4	0	9	58	0	10	0				599	1,41	0,24	598
5	0	12	13	0	12	13				733	0,00	0,00	733
6	0	14	46	0	14	47				887	0,71	0,08	886
7	0	16	30	0	16	28				989	1,41	0,14	990
8	0	19	35	0	19	34				1.175	0,71	0,06	1.175
9	0	21	14	0	21	14				1.274	0,00	0,00	1.274
10	0	23	43	0	23	40				1.422	2,12	0,15	1.423
11	0	25	46	0	25	46				1.546	0,00	0,00	1.546
12	0	28	14	0	28	13				1.694	0,71	0,04	1.694
13	0	30	13	0	30	11				1.812	1,41	0,08	1.813
14	0	32	53	0	32	54				1.974	0,71	0,04	1.973
15	0	34	38	0	34	37				2.078	0,71	0,03	2.078
16	0	37	12	0	37	10				2.231	1,41	0,06	2.232
17	0	39	15	0	39	16				2.356	0,71	0,03	2.355
18	0	41	38	0	41	37				2.498	0,71	0,03	2.498
19	0	43	48	0	43	47				2.628	0,71	0,03	2.628
20	0	46	18	0	46	16				2.777	1,41	0,05	2.778
21	0	48	25	0	48	24				2.905	0,71	0,02	2.905
22	0	50	49	0	50	43				3.046	4,24	0,14	3.049
23	0	53	9	0	53	8				3.189	0,71	0,02	3.189
24	0	55	36	0	55	34				3.335	1,41	0,04	3.336
25	0	57	57	0	57	56				3.477	0,71	0,02	3.477
26	1	0	36	1	0	35				3.636	0,71	0,02	3.636
27	1	2	40	1	2	39				3.760	0,71	0,02	3.760
28	1	5	15	1	5	13				3.914	1,41	0,04	3.915
29	1	7	33	1	7	30				4.052	2,12	0,05	4053
SF	0	7	37	0	7	36				457	0,71	0,15	457
Total													61.479