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

TIME ASSÍNCRONO INICIALIZADOR PARA O PLANEJAMENTO DA EXPANSÃO DA TRANSMISSÃO

Claudio Renato Thomas de Almeida

Orientador: Prof. Dr. Alcir José Monticelli

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

Campinas, Junho de 1998.

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

AL64t

Almeida, Claudio Renato Thomas de
Time assíncrono inicializador para o planejamento da expansão da transmissão. / Claudio Renato Thomas de Almeida.--Campinas, SP: [s.n.], 1998.

Orientador: Alcir José Monticelli
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento paralelo (Computadores). 2. Programação paralela (Computação). 3. Energia elétrica - Transmissão. 4. Energia elétrica – Planejamento. I. Monticelli, Alcir José. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

aos meus pais,

Yvone e Claudio.

Agradecimentos

Em primeiro lugar, sou grato a Deus por tudo. Graças a Ele sou o que sou e cheguei onde cheguei. A começar pela oportunidade de ter os pais maravilhosos que tenho, que me deram todas as condições para que eu alcançasse todos os meus objetivos, além é claro, de todo o apoio, incentivo e orientação sempre oportunos e benvindos.

Obrigado às minhas amigas Mara de Lucio (HC) e Sonia Campos (FEQ) que me ajudaram muito com todo o apoio nos momentos pessoais mais difíceis.

À minha querida amiga Sonia, obrigado por todo o carinho, compreensão, incentivo e paciência dados durante a fase mais difícil do trabalho.

Sou grato ao Prof. Monticelli pela oportunidade de ser seu orientado. Sou uma pessoa de sorte, por ter a honra de ver o meu nome ao lado do nome de um profissional com tamanho gabarito, competência e reconhecimento no meio acadêmico.

À minha adorada Lê, e à minha amiga Elisa (sua irmã) que apareceram em minha vida já na reta final deste trabalho, mas cujas presenças serviram como mais um estímulo para que o trabalho fosse concluído com êxito.

Finalmente, quero agradecer a todos os amigos do DSEE, em especial ao Prof. (e futuro Doutor) Sergio Azevedo de Oliveira, não só pelo incentivo, pela colaboração nos trabalhos, pela paciência gasta comigo, mas principalmente pela amizade, que foi fundamental para o sucesso dos trabalhos e nosso crescimento como indivíduos.

Este trabalho contou com o apoio financeiro da Fundação CAPES (Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e com o patrocínio familiar do Sr. Claudio Viana de Almeida e D. Yvone de Almeida.

Claudio R.T. de Almeida.

Lista de Abreviaturas e Símbolos

Palavras em *itálico* - expressões em outro idioma que possuem correspondente em português, ou palavras destacadas. Expressões em outro idioma que não possuem um correspondente em português realmente utilizado, ou mesmo porque são palavras consagradas pelo uso não serão apresentadas em itálico.

Palavras entre aspas - palavras em sentido figurado, ou nomes próprios.

Time-A - Time Assíncrono (do inglês, *A-Team*).

PVM - Parallel Virtual Machine. Software e conjunto de bibliotecas destinados a programação para Processamento Paralelo.

TSP - Traveling Salesman Problem. Problema do Caixeiro Viajante.

MAP - Multi-Algorithm Problem. Problema Multi-Algoritmo.

FCO - Fluxo de Carga Ótimo.

PNLIM - Problema de Programação Não Linear Inteiro Misto.

Solaris - Sistema Operacional UNIX desenvolvido pela Sun Microsystems.

MINOS - Software destinado a resolução de problemas de otimização.

SPARC - Nome básico dos modelos de estações de trabalho produzidas pela SUN.

MVP - Máquina Virtual Paralela.

daemon - Processo responsável pelo gerenciamento de uma determinada tarefa do sistema.

FIFO - *First-In-First-Out*, representa um tipo de armazenamento de dados, no qual o primeiro elemento a ser armazenado necessariamente será o primeiro a ser lido ("retirado da pilha"), e assim por diante.

ID - Número de Identificação.

Resumo

Este trabalho apresenta uma nova abordagem do inicializador para o planejamento da expansão de sistemas de transmissão de energia elétrica, utilizando-se o modelo de Times Assíncronos (*Times-A*).

Nenhum dos algoritmos para otimização com satisfação de restrições funciona sem fragilidade. Esses algoritmos tendem a ser lentos, e as heurísticas pouco confiáveis. É por isso que, ao invés de se procurar algoritmos novos e melhores , tem-se experimentado meios de usar os já existentes em conjunto, de tal forma que eles consigam fazer o que separadamente não conseguem. Essa é a idéia do que se denomina *Time Assíncrono* , que normalmente combina implementações de vários métodos heurísticos, na busca das melhores soluções possíveis para problemas complexos.

Uma parte importante do processo de planejamento é a determinação de famílias de soluções iniciais contendo características atrativas. Essas soluções em geral contém linhas e conjuntos de linhas (blocos construtivos) que aparecerão mais tarde nas soluções ótimas obtidas via métodos como os Algoritmos Genéticos, *Simulated Annealing* e Busca Tabu. Acredita-se que uma paralelização destes métodos via Times Assíncronos poderá ser altamente eficaz. Assim, neste trabalho buscou-se caminhar nessa direção. Como um primeiro passo, foi implementada uma versão de Times-A para resolver o problema de inicialização do problema de planejamento estático, utilizando-se os métodos heurísticos aproximados de Garver, Mínimo Esforço e Mínimo Corte de Carga.

Testes foram realizados em uma rede heterogênea de estações para sistemas de pequeno, médio e grande porte. O processamento paralelo do sistema foi implementado através do software PVM (*Parallel Virtual Machine*), e as primitivas correspondentes utilizadas na programação são apresentadas nos apêndices.

Palavras Chaves: *Times Assíncronos, Processamento Distribuído, Métodos Heurísticos, Planejamento da Transmissão.*

Sumário

Capítulo 1

Introdução

1.1 O Inicializador e o Problema do Planejamento

O planejamento da expansão de redes de transmissão de sistemas de energia elétrica determina *quando*, *onde* e *que* tipos de linhas e/ou transformadores devem ser instalados na rede a fim de que o sistema opere adequadamente para uma demanda futura predeterminada e realizando o menor investimento possível. O planejamento *dinâmico* (*quando*) em geral é decomposto em subproblemas *estáticos* que tratam das questões *onde* e *que tipo* (planejamento em um estágio, de um ano inicial a um ano final, dados).

Este trabalho trata especificamente da fase estática do planejamento da transmissão. Métodos tipo Algoritmos Genéticos e Busca Tabu normalmente necessitam da determinação prévia de blocos construtivos atraentes (conjuntos coerentes de linhas e transformadores). Isto exige uma inicialização adequada desses métodos. A parte principal tanto dos Algoritmos Genéticos como dos métodos baseados na Busca Tabu são facilmente paralelizáveis. Acredita-se que uma paralelização via Times Assíncronos poderá ser altamente eficaz. Assim, neste trabalho buscou-se caminhar nessa direção. Como um primeiro passo, foi implementada uma versão de Times-A para resolver o problema de inicialização do planejamento estático, objetivando redução de tempo de processamento e diversificação das soluções iniciais. Este Time-A implementado será denominado *Inicializador*. Desta forma, este trabalho é uma parte de um projeto mais amplo, que já estava em desenvolvimento na Unicamp, onde os algoritmos combinatoriais citados serão utilizados como agentes em Times-A para a resolução do problema do planejamento da expansão da transmissão de sistemas de grande porte, como o Norte-Nordeste Brasileiro de (89 barras - 179 linhas).

Em uma época na qual os recursos computacionais eram mais limitados, foram desenvolvidos métodos heurísticos como por exemplo os métodos de Garver [GARV70], Mínimo Esforço

[MONT82] e Mínimo Corte de Carga [PERE85a]. Essas metodologias ainda são utilizadas por concessionárias como parte de procedimentos interativos que exigem uma participação ativa dos planejadores. Este trabalho incorpora essas metodologias na construção dos Times-A inicializadores.

Utilizou-se um modelo de programação multi-agente conhecido como Times Assíncronos (*Times-A*). Um Time-A consiste de uma rede computacional fortemente cíclica, onde agentes autônomos cooperam, trabalhando todo o tempo em paralelo, com o compartilhamento constante de resultados entre os membros do time. Inicialmente desenvolveu-se um *Time-A* com variantes dos algoritmos baseados no método de Garver, e, posteriormente, um outro time acrescido de variantes dos algoritmos de Mínimo Esforço e Mínimo Corte de Carga.

1.2 Modelagem e Resolução do Problema do Planejamento da Transmissão

1.2.1 Modelagem do Problema

Modelo DC

O problema mostrado a seguir (1.1) possui formulação do tipo PNLIM e pertence ao conjunto de problemas chamados *NP-completo*, que são de difícil tratamento, e que apresentam o problema da explosão combinatorial pois geralmente existem muitos caminhos candidatos e, além disso, em cada caminho podem ser alocadas várias linhas. Porém, quando os valores de x_{ij} são especificados, ou seja, quando é determinada uma proposta de investimento ou configuração, a formulação 1.1 se reduz a um simples problema de programação linear onde a única finalidade é verificar a factibilidade da proposta de investimento, isto é, verificar se o investimento proposto produz um corte de carga nulo.

O problema 1.1, chamado de Modelo DC, é considerado como o modelo ideal para ser usado em planejamento de sistemas de transmissão. Este modelo somente leva em conta as duas leis de Kirchhoff para o sistema elétrico e a capacidade de transmissão das linhas existentes e das candidatas.

$$
\begin{aligned}
Min \quad & v = \sum_{(i,j)\in\Omega} c_{ij} n_{ij} \quad + \quad \sum_{i} \alpha_i r_i \qquad (1.1)\\
s.a. \quad & \\
& B(x+\gamma^0)\theta + g + r = d \\
& (x_{ij}+\gamma^0_{ij})|\theta_i - \theta_j| \leq (x_{ij}+\gamma^0_{ij})\overline{\phi}_{ij} \\
& 0 \leq g \leq \overline{g} \\
& 0 \leq r \leq d \\
& 0 \leq n_{ij} \leq \overline{n}_{ij} \\
& \forall (i,j) \in \Omega
\end{aligned}
$$

c_{ij} Custo do ramo (i,j).

x_{ij} Nova susceptância a ser instalada no ramo $(i,j) \in \Omega$.

$B(.)$ Matriz de susceptâncias.

θ Ângulos das tensões nodais.

γ_{ij}^0 Susceptância inicial no ramo $(i,j) \in \Omega$.

n_{ij} Número de linhas adicionadas, $n_{ij} = \frac{x_{ij}}{\gamma_{ij}}$, sendo γ_{ij} a susceptância nominal de uma linha entre as barras i-j.

$\overline{\phi}_{ij}$ Definida pela relação: $\overline{\phi}_{ij} = \frac{\overline{f_{ij}}}{\gamma_{ij}^0}$, sendo $\overline{f_{ij}}$ o fluxo máximo de potência permitido que passa pela linha i-j.

Ω Conjunto de todos os ramos definidos pelas linhas existentes e as alternativas de expansão.

g Vetor de gerações.

d Vetor de cargas.

$\overline{g}$ Vetor de limites de geração.

r Vetor de geradores fictícios ou artificiais.

α Parâmetro adequado de transformação de unidades.

O Modelo de Transportes

O Modelo de Transportes (1.2) foi formulado por Garver em [GARV70] e teve muito sucesso, sendo esta a primeira proposta para planejamento de redes de transmissão que usou programação linear. Este método consiste basicamente em resolver de maneira aproximada uma versão relaxada do Modelo DC. No modelo de Garver, conhecido como *Modelo de Transportes*, somente se leva em conta a Primeira Lei de Kirchhoff e a capacidade de transmissão das linhas, não sendo levado em conta, portanto, a Segunda Lei de Kirchhoff.

$$
\begin{aligned}
Min \quad & v = \sum_{(i,j)\in\Omega} c_{ij} n_{ij} \quad + \quad \sum_i \alpha_i r_i \qquad (1.2)\\
s.a. \quad & \\
& Sf + g + r = d \\
& |f_{ij}| - (x_{ij} + \gamma_{ij}^0)\overline{\phi}_{ij} \leq 0 \\
& 0 \leq g \leq \overline{g} \\
& 0 \leq r \leq d \\
& 0 \leq n_{ij} \leq \overline{n}_{ij} \\
& \forall (i,j) \in \Omega
\end{aligned}
$$

onde S é a matriz de incidência nó-ramo e f_{ij} o fluxo de potência no ramo i-j.

O Modelo Híbrido

Este modelo (1.3) é uma combinação do Modelo de Transportes e do Modelo DC. Neste, trata-se de contornar as desvantagens na solução do Modelo de Transportes. Adicionou-se ao mesmo somente uma parcela das restrições correspondentes à Segunda Lei de Kirchhoff, isto é, são consideradas como parte da formulação aquelas restrições da Segunda Lei correspondentes às linhas existentes e eliminadas as restrições correspondentes aos novos caminhos. Essa nova formulação também foi introduzida por Garver em [VILL85].

$$
\begin{aligned}
Min \quad & v = \sum_{(i,j)\in\Omega} c_{ij} n_{ij} \quad + \quad \sum_{i} \alpha_i r_i \qquad (1.3)\\
s.a. \quad & \\
& B_1\theta + S_2 f + g + r = d \\
& |\theta_i - \theta_j| \leq \overline{\phi}_{ij} \qquad \forall (i,j) \in \Omega_1 \\
& |f_{ij}| \leq (x_{ij} + \gamma_{ij}^0)\overline{\phi}_{ij} \qquad \forall (i,j) \in \Omega_2 \\
& 0 \leq g \leq \overline{g} \\
& 0 \leq r \leq d \\
& 0 \leq n_{ij} \leq \overline{n}_{ij}
\end{aligned}
$$

onde

- Ω_1 Representa o conjunto de ramos onde já existem linhas na configuração básica.
- Ω_2 Representa o conjunto de ramos candidatos em caminhos onde não existe linha na configuração inicial.
- B_1 É a matriz de susceptância para os ramos que pertencem a Ω_1.
- S_2 É a matriz de incidência nó-ramo para os ramos que pertencem a Ω_2.

1.2.2 Algoritmos Heurísticos Construtivos

Metodologia de Garver

O algoritmo de Garver [GARV70], é um processo chamado de passo-a-passo onde em cada iteração do algoritmo se toma uma decisão de adicionar uma linha à configuração atual. A linha adicionada é aquela que aparece mais sobrecarregada quando o Modelo de Transportes 1.2 é resolvido.

Metodologia de Villanasa-Garver

A metodologia proposta por Villanasa [VILL85] é uma extensão da metodologia de Garver, na qual se adiciona a Segunda Lei de Kirchhoff na rede existente na formulação apresentada em 1.3.

Metodologia do Mínimo Esforço

Este método foi desenvolvido na Unicamp e é formulado usando o Modelo DC. Assim como Garver, este também faz um plano de expansão passo-a-passo, isto é, para uma configuração da rede os circuitos são adicionados um a um em pequenos grupos. O critério para a adição do próximo circuito é determinado por uma análise de sensibilidade, chamada de mínimo esforço.

Metodologia de Mínimo Corte de Carga

Este algoritmo [PERE85a], usado no planejamento de sistemas de transmissão, funciona de maneira que em cada passo do mesmo é produzida uma adição de um circuito na configuração base, até que o sistema opere adequadamente, ou seja, sem corte de carga. Este circuito é selecionado de acordo com um índice de desempenho ou índice de sensibilidade.

1.2.3 Métodos de Otimização Clássica

Estes métodos fazem parte de uma nova fase da resolução do planejamento da expansão, na tentativa de resolver a formulação 1.1 de maneira ótima e a principal ferramenta encontrada foram as técnicas de decomposição matemática. Nesta perspectiva, a metodologia mais usada foi a *Técnica de Decomposição de Benders*, a qual explora a decomposição natural do problema do planejamento da expansão de sistemas de transmissão em duas partes : subproblema de investimento e subproblema de operação.
Os primeiros pesquisadores a empregar os esquemas de decomposição matemática foram os do CEPEL [PERE85b][PERE87], onde a decomposição de Benders é utilizada com o Modelo de Transportes e DC separadamente. O mesmo grupo apresenta um trabalho posterior [GRAN85], onde é realizada uma análise teórica sobre os problemas de convexidade e sobre as características dos cortes de Benders.
Em [ROME94], o grupo da Unicamp propõe o uso de um esquema de decomposição hierarquizado em três partes.
Posteriormente aparecem outros trabalhos como [OLIV95] onde também há um esquema de decomposição hierarquizado, porém em duas fases. Finalmente em [BAYO94] é proposto um método heurístico que aproveita a decomposição natural do problema em subproblemas de operação e investimento.

1.2.4 Métodos de Otimização Combinatorial

A resolução de um problema de otimização combinatorial consiste em se determinar a solução ótima entre um grande número de soluções alternativas. Assume-se para isso que a qualidade de uma solução é quantificável e que pode ser comparada com uma outra solução, além do fato do conjunto de soluções ser finito. Estes problemas, que possuem a propriedade *NP-completo*, podem ser resolvidos por vários algoritmos, que por sua vez podem ser selecionados entre duas opções : uma que obtém o ótimo global com a desvantagem de exigir tempos de computação inviáveis, e outra que retorna soluções sub-ótimas com menor tempo de processamento.
Dentro da primeira opção, que constitui a classe de algoritmos de otimização, há exemplos bem conhecidos como o método *Branch and Bound*, um método de enumeração que usa planos de corte . A segunda opção, que constitui a classe de algoritmos de aproximação, também conhecidos como algoritmos heurísticos, envolve vários métodos muito utilizados como *Simulated Annealing*, Algoritmos Genéticos e Busca Tabu.

1.3 A Natureza da Organização

A idéia de reunir ou combinar diferentes esforços na tentativa de se alcançar um determinado objetivo é inerente aos seres vivos, como demonstram muitos exemplos do mundo natural e da história. Na sociedade contemporânea aplicam-se esses princípios básicos de organização em métodos, processos ou sistemas nas mais diversas áreas de trabalho. Na indústria por exemplo, as *linhas de montagem* idealizadas por Henry Ford são um exemplo de como tarefas organizadas trabalhando em paralelo aumentam a produtividade através da redução do tempo total da produção. Na medicina, "coquetéis" de drogas são utilizados na tentativa de se obter melhores resultados no tratamento de doenças como a AIDS, sempre visando menor sofrimento e vida longa ao ser humano.

Na área acadêmica, graças à constante evolução das tecnologias de hardware e software, pesquisadores têm tido a oportunidade de criar e testar diversos modelos computacionais sempre com o intuito de se otimizar um determinado sistema, seja através da redução do tempo total de processamento, seja através de soluções de menor custo. Para isso, vários pesquisadores vêm utilizando já há algum tempo os conceitos de *Processamento Paralelo* e *Organizações Multi-Agente*, ou mais especificamente, *Times Assíncronos*.

1.4 Estado-da-Arte

A seguir, são relacionados alguns trabalhos que enfocam o uso de organizações multi-agente, sua relação com o processamento paralelo, bem como os diversos métodos que podem ser combinados na resolução de vários problemas.

- Mesmo considerando implementações num ambiente seqüencial, Ortega e Rheinboldt apresentam em 1968 [ORTE68] um trabalho em que se estuda a convergência de métodos iterativos combinados.

- Em problemas de engenharia, Dusonchet *et al.* [DUSO71] , propõem a solução do problema de Fluxo de Potência combinando-se os métodos de Jacobi para as barras de carga e o de Newton para as barras de tensão controlada, obtendo-se resultados significativos.

- A partir da década de 80 os *Team Algorithms* começaram a ser estruturados como uma combinação de diferentes métodos num sistema distribuído assíncrono. A primeira referência existente é o trabalho de Talukdar, Pyo e Giras [TALU83a], e outro em seguida, de Talukdar, Pyo e Mehrotra [TALU83b]. Nesses trabalhos citados, a combinação de algoritmos é feita sob a supervisão de um *Administrador* que utiliza uma memória global, conhecida como *Blackboard.*

- Num trabalho de 1983, Mehrotra e Talukdar [MEHR83] apresentam uma ferramenta que, dado um conjunto de algoritmos divididos em tarefas, distribui as mesmas entre uma série de processadores e sugere as mudanças necessárias para a redução do tempo total de execução.

- Em 1985, Pyo [PYO 85] apresentou um trabalho que foi a inspiração para a criação dos Times Assíncronos, onde ele chama de *team approach* os meios de se montar um algoritmo assíncrono distribuído através da combinação de algoritmos já existentes. Neste trabalho, Pyo resolveu equações algébricas aplicando os métodos Newton-Raphson (NR) e Steepest Descent (SD).

- Também em 1985, temos um outro trabalho, de Talukdar e Cardozo [TALU85], onde são enfatizadas as vantagens da utilização de um *Team Algorithm* com um administrador inteligente.

- Em 1990, Talukdar e de Souza [TALU90] introduziram o termo *Times Assíncronos* apresentando um time de agentes assíncronos também para a resolução de equações algébricas não-lineares.

- E em 1991, de Souza e Talukdar [DSOU91] apresentam combinações de algoritmos genéticos (GA), com métodos já tradicionais como os de Newton e o Levenberg-Marquardt, daí então denominando Times-A (Times Assíncronos) estas combinações de métodos em sistemas distribuídos assíncronos, sendo que nesta proposta cada processador trabalha usando seu próprio método.

- No trabalho de Talukdar, Ramesh e Nixon [TALU91], é feita uma analogia entre os Times-A e uma sociedade de insetos.

- Em Ramesh *et al.* [RAME91], de 1991, os Times-A são utilizados em aplicações diversas como: solução de sistemas de equações não-lineares, combinação de Algoritmos Genéticos com algoritmos de cálculo intensivo, projeto de edifícios, controle de redes elétricas e outras aplicações em sistemas de potência.

- Em 1992, Talukdar, Ramesh, Quadrel e Christie [TALU92a] discutem a presença de sistemas multi-agente nas operações em tempo de real de sistemas elétricos de potência, descrevendo os mesmos em termos de sua estrutura e comportamento.

- Resultados promissores através do uso de Times Assíncronos são encontrados no trabalho de Talukdar e de Souza de 1992 [TALU92b], principalmente com relação ao problema do caixeiro viajante (TSP), onde o Time-A organizado consegue superar todos os melhores algoritmos hoje disponíveis para os maiores problemas de solução exata conhecida (318 e 532 cidades).

- Em 1993, Talukdar e Ramesh [TALU93b] propuseram uma nova abordagem baseada em Times Assíncronos para resolver o problema de como atualizar planos para corrigir um dado conjunto de contingências em uma rede de potência. Sua abordagem decompõe o problema em um conjunto de problemas fracamente acoplados e muito pequenos que podem ser resolvidos por um time de agentes cooperativos.

- de Souza [DSOU93] propõe uma modelagem de Times Assíncronos para a resolução de *Problemas Multi-Algoritmo* (*MAPs*), mais especificamente para a resolução do *Problema do Caixeiro Viajante* (*TSP Problem*).

- Times Assíncronos são utilizados por Talukdar e Ramesh [TALU94] na resolução do problema do fluxo de carga ótimo, onde este é dividido em subproblemas menores que são tratados em paralelo.

- Um modelo de programação assíncrona é proposto por Rodrigues, Saavedra e Monticelli [RODR94] para a resolução do FCO com restrições de segurança que permite o desenvolvimento de aplicações que podem ser portadas entre diferentes arquiteturas de computadores paralelos de maneira transparente e sem perda significativa de eficiência.

- Em 1996 Ramesh e Talukdar [RAME96] apresentam um trabalho onde efetuam uma decomposição assíncrona da lista de contingências de um FCO, de tal forma que estas sejam avaliadas em paralelo para que o tempo total de processamento seja o mesmo para o cálculo de um FCO sem qualquer contingência.

1.5 Composição do Trabalho

Este trabalho encontra-se estruturado da seguinte forma: no capítulo 2, são detalhados os conceitos e teorias necessários para a compreensão da modelagem de um time assíncrono; no capítulo 3, é feita uma breve apresentação teórica dos Métodos Heurísticos Construtivos utilizados como agentes (*Garver* [GARV70], *Mínimo Esforço* [MONT82] e *Mínimo corte de Carga* [PERE85a]), componentes do time implementado; no capítulo 4, são especificados todos os detalhes da estrutura e do funcionamento do time assíncrono proposto para o Inicializador; no capítulo 5, são exibidos e discutidos os resultados dos vários testes realizados para várias configurações do time para os três sistemas elétricos considerados; no capítulo 6, são apresentadas as conclusões gerais.

Ao final do trabalho, dois apêndices contém informações a respeito das primitivas do PVM utilizadas nos programas (Apêndice A) e informações sobre os sistemas de potência considerados nos testes (Apêndice B).

Foram feitos testes para três sistemas: Garver (06 barras - 15 linhas), Sul brasileiro (46 barras - 79 linhas) e Norte-Nordeste Brasileiro (89 barras - 179 linhas), em uma rede homogênea de estações SUN, sistema operacional Solaris 2.5.1, utilizando-se o software PVM 3.3.5 [GEIS94].

Capítulo 2

Times Assíncronos

2.1 Introdução

Neste capítulo, serão relatados os principais conceitos, fundamentos e características que envolvem a teoria de Times Assíncronos, de acordo com o De Souza [DSOU93], onde o problema central é a resolução do TSP Euclideano. Mais adiante no capítulo 4 serão detalhados os mesmos conceitos porém com respeito ao modelo criado para a inicialização do problema do planejamento da expansão da transmissão de sistemas de potência, tema central deste trabalho.

2.2 Classificação de Problemas e Modelos de Organização

2.2.1 Espaço de Problemas

Segundo De Souza [DSOU93], problemas podem ser classificados em três categorias gerais, como se vê na Fig. 2.1. Cada categoria representa os problemas de acordo com os algoritmos disponíveis para resolvê-los. Desta forma, temos a seguinte classificação :

1. Algoritmos Adequados ;
2. Algoritmos Inadequados ;
3. Algoritmos Inaptos ;

A primeira categoria, *Algoritmos Adequados*, engloba os problemas com pelo menos um algoritmo adequado capaz de resolvê-los. Um exemplo é um problema de Programação Inteira sendo

resolvido pelo algoritmo *branch-and-bound.*

A segunda categoria, denominada *Algoritmos Inadequados*, pode ser subdividida em dois tipos de problemas. O primeiro se refere a problemas que pedem a melhor solução possível dentro de uma certa quantidade de tempo. Um exemplo típico são os problemas de otimização. Se os algoritmos disponíveis fornecessem o ótimo global, então o problema poderia pertencer à primeira categoria (Algoritmos Adequados), mas na maioria das vezes os algoritmos disponíveis podem apenas gerar soluções aproximadas, apesar de factíveis. O segundo tipo de problemas nesta categoria, envolve aqueles cujos algoritmos disponíveis geram soluções com alguma violação de restrição; porém, a região de soluções geradas por estes algoritmos não é tão longe da região factível.

Finalmente, a terceira categoria, Algoritmos Inaptos, envolve problemas cujos algoritmos disponíveis falham inclusive em fornecer soluções factíveis. Em contraste com a segunda categoria, os algoritmos disponíveis para estes tipos de problemas violam grosseiramente as restrições do problema, consequentemente gerando uma região de soluções muito distante da região factível. Problemas nesta categoria são muito difíceis ou tem restrições muito "apertadas", ou seja, o tempo necessário para a resolução do problema com estas restrições é muito grande.

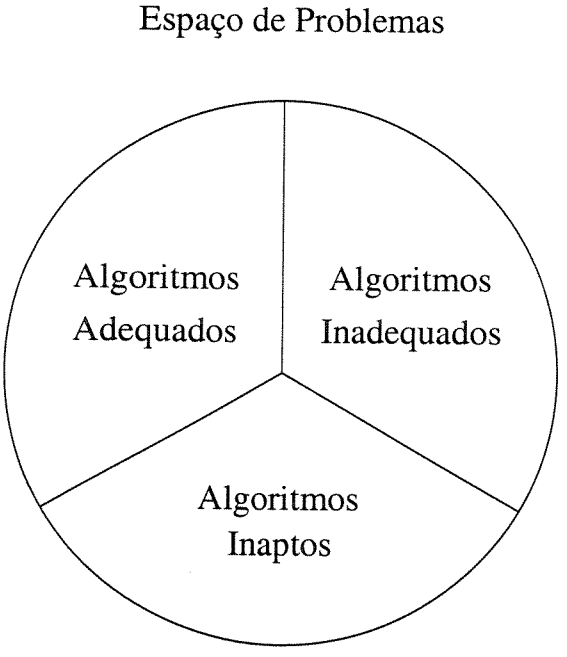


Figura 2.1: Espaço de Problemas caracterizado pelo número de algoritmos disponíveis para a resolução de um dado problema.

2.2.2 Problemas Multi-Algoritmo

Um problema pode ser decomposto em três espaços de variáveis [DSOU93]:

1. Espaço das variáveis de entrada : contém todas as variáveis que definem o problema. Normalmente há apenas um elemento que especifica o caso de um problema a ser resolvido.
2. Espaço das variáveis de saída : contém todas as variáveis que especificam soluções para um problema.
3. Espaço das variáveis de estado : contém todas as variáveis que são de interesse durante o processo de resolução do problema, mas que não constituem soluções factíveis.

Na Fig. 2.2 temos um diagrama que ilustra essa decomposição do problema. Os espaços são conectados por setas que representam somente os algoritmos disponíveis para um problema, que mapeiam elementos de um espaço em outro.

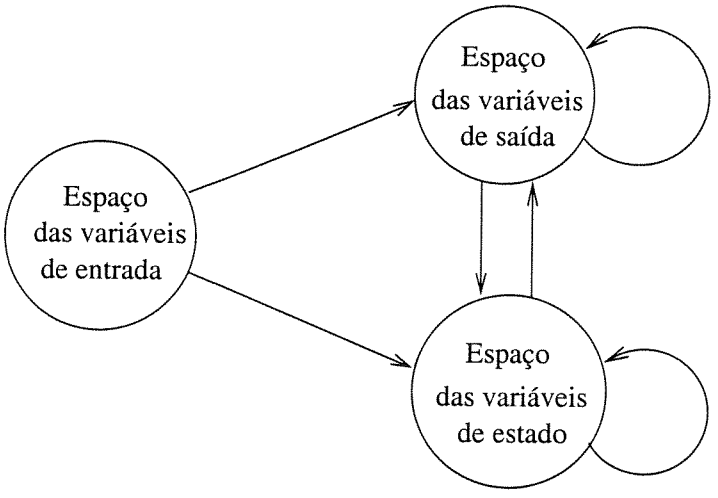


Figura 2.2: Diagrama de Decomposição de Problema.

Um Problema Multi-Algoritmo (*Multi-Algorithm Problem*, ou *MAP*) [DSOU93], é um problema que pertence à categoria dos Algoritmos Inadequados e possui pelo menos um ciclo no seu diagrama de decomposição do problema. A presença de ao menos um ciclo garante que lá existe um ou mais algoritmos que podem ser executados iterativamente, que é a característica usada pelo método de solução proposto por de Souza em sua tese.

Dependendo da abordagem considerada para a resolução de um dado problema, pode-se criar diversos algoritmos distintos. Estes, por sua vez, podem diferenciar em complexidade, eficiência,

robustez e muitas outras características, sendo que cada um deles possui seus pontos fortes e fracos. Por um lado, pontos fracos de um algoritmo podem ser sobrepostos pelos pontos fortes de outros, e pontos fortes podem beneficiar uns aos outros. Por outro lado, deve-se evitar a situação na qual pontos fracos são agrupados juntos e pontos fortes aniquilam uns aos outros. Mas informações precisas sobre os algoritmos e como eles devem ser combinados são raramente disponíveis.

Exemplo de MAP: versão Euclidiana do Problema do Caixeiro Viajante (TSP). Este é um MAP porque não há nenhum algoritmo que possa resolver um caso grande arbitrário deste problema num intervalo razoável de tempo, e também por haver um número grande de heurísticas que dão soluções aproximadas para este problema.

2.2.3 Modelos de Organização

Eis aqui uma breve apresentação dos modelos de organização encontrados na literatura. Isto se faz necessário, uma vez que o conceito de Times Assíncronos é um passo inovador no domínio da Teoria da Organização [SHAF87].

Neurônios e o Cérebro Humano

O cérebro humano pode ser dividido em cinco áreas: *cortex*, *cerebellum*, *midbrain*, *pons* e *medulla*. Todos juntos comandam as funções do organismo como os movimentos, a visão, a audição, etc. O cérebro humano é composto por células chamadas de neurônios, que são responsáveis pela transmissão das informações. Estes, por sua vez, são formados por dendritos, que recebem as informações, e um axônio, que transmite as informações para o neurônio seguinte. Apesar da complexidade dessa enorme rede de neurônios e das tarefas executadas pelo cérebro, não há nenhum controle central conhecido entre essas células.

A organização do cérebro tem sido estudada sob vários aspectos. Estes estudos têm sido aplicados com sucesso a problemas de engenharia através da reprodução aproximada das funções e capacidades de aprendizado do cérebro para a resolução de problemas. Por exemplo, redes neurais tem sido aplicadas com sucesso em aplicações de reconhecimento de imagens.

Sociedade de Insetos

Castas e divisão do trabalho são as características marcantes na organização de uma sociedade de insetos. Dentre as milhares de espécies de sociedades de insetos, há três características comuns a todas, como coloca Oster *et al.* [OSTE78] :

1. cooperação no sentido de se cuidar dos jovens ;
2. superposição de pelo menos duas gerações capazes de contribuir para o trabalho colonial ;

3. divisão reprodutiva do trabalho com a maior parte ou a totalidade dos operários estéreis trabalhando a favor dos indivíduos encarregados da reprodução ;

A capacidade de executar tarefas simultaneamente é o maior fator que torna uma colônia ou sociedade mais eficiente que um indivíduo independente forte. Graças ao trabalho em paralelo, uma colônia pode reagir imediatamente diante de diversas situações, como diante de um inimigo, um pedido de ajuda, ou qualquer fator inesperado. Mais ainda, devido à existência de múltiplos indivíduos capazes de executar a mesma tarefa, a falha de um individualmente é reparada com sucesso pelos outros, o que aumenta muito a confiabilidade de uma colônia no desempenho de tarefas se comparada com a confiabilidade de um indivíduo independente.

Organizações Humanas

Quando se fala em organização humana, pensa-se em um grupo de indivíduos inter-relacionados que possuem objetivos comuns, e cujas relações obedecem uma certa estrutura. O estudo das organizações humanas tornou-se importante desde o começo do século XX, com o advento das grandes corporações. Desde então, alguns dos modelos organizacionais tem sido aplicados não somente em corporações de negócios, mas também em outros tipos de organizações. Fox [FOX 79], define algumas estruturas básicas de organização humanas :

1. Pessoa Simples : somente uma pessoa executa todas as tarefas necessárias para se atingir um objetivo desejado.

2. Grupo : é necessário mais do que uma pessoa para que tarefas mais complexas sejam finalizadas. Subtarefas são designadas a cada indivíduo que é coordenado em direção a um objetivo compartilhado. Coordenação é obtida através de compartilhamento de toda a informação e da adoção de uma solução que satisfaça todos os membros.

3. Hierarquia Simples : em grandes grupos, torna-se difícil a obtenção de uma solução que satisfaça todos os membros. Daí, uma tarefa especial de decisão é designada a um único membro, sendo que só este tem acesso a toda informação, reduzindo-se assim os custos com comunicação.

4. Hierarquia Uniforme : consiste num gerenciamento de nível múltiplo, criado nos casos em que o grupo cresce muito de tal forma que torna-se impossível para apenas um membro tomar todas as decisões.

5. Hierarquia Multi-Divisional : a medida que a complexidade e o número de objetivos cresce, a comunicação e o controle tornam-se inapropriados até para hierarquias uniformes. Daí, estas são repartidas em divisões de acordo com seus objetivos, criando uma hierarquia multi-divisional.

τ-net : um Modelo Organizacional

Talukdar e Souza [TALU93a] definem τ-net como sendo um modelo de rede de super-agentes. Seu propósito é ajudar a visualização da estrutura destes super-agentes. Esta estrutura por sua vez, é definida pelo modo como os componentes (memórias e processos) dos super-agentes estão conectados. τ-net é baseada nas seguintes considerações:

- serão considerados apenas os super-agentes cujos correspondentes sistemas de comunicações podem ser modelados por conjuntos de memórias compartilhadas.
- os estados internos dos agentes não são de interesse.

Pode-se imaginar um super-agente como sendo uma coleção de memórias que são compartilhadas pelos seus agentes. Objetos são transformados uns em outros diferentes, uma vez que os agentes lêem e escrevem nas memórias compartilhadas. A estrutura de um super-agente é representada por dois fluxos: fluxo de dados e fluxo de controle.

O fluxo de dados determina quais memórias cada agente pode acessar. O fluxo de controle define que dados devem ser escolhidos nas memórias (política de seleção de entrada), quando um agente irá atuar (planejamento), e que recursos alocar (política de alocação de recursos).

A taxonomia τ-net é utilizada para apresentar algumas organizações de software, como pode se ver na Fig. 2.3. A primeira coluna representa o par: tipo do fluxo de dados (linhas pretas) e o fluxo de controle (linhas pontilhadas) para cada estrutura de organização na segunda coluna. Na terceira coluna são apresentados exemplos de organizações que possuem tais estruturas.

A expressão $T(\tau)$ representa o espaço de design para super-agentes computacionais, isto é, $T(\tau)$ é o espaço que contém todas as τ-nets.

Como pode se ver na Fig. 2.3, um *Time-A* possui fluxo de dados cíclico e fluxo de controle nulo, e difere de uma organização de software comum chamada *Blackboards*, uma vez que estes possuem fluxo de dados cíclico e fluxo de controle acíclico.

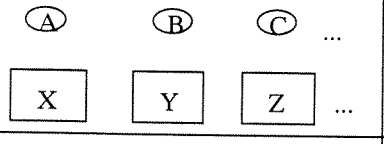
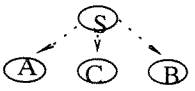
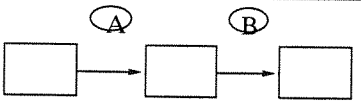
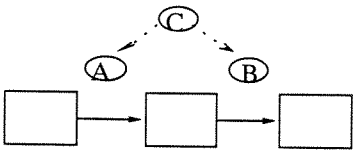
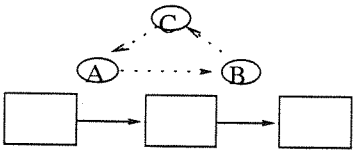
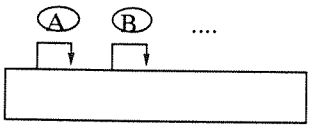
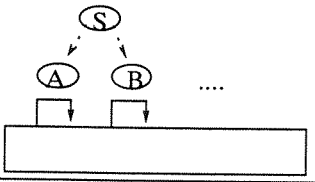
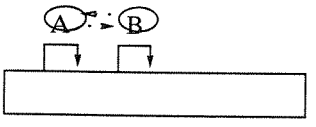
Região de T(τ)	Um exemplo típico	Aplicações Computacionais
[Nulo,Nulo]		Uma biblioteca, isto é, um conjunto de bancos de dados e conjuntos desconectados.
[Nulo,Acíclico]		Agentes que compartilham computadores, mas não dados. O fluxo de controle é para alocação de computadores.
[Nulo,Cíclico]		Normalmente não utilizado.
[Acíclico,Nulo]		Um conjunto autônomo de agentes conectados *head to toe* .
[Acíclico,Acíclico]		Configuração mais comum.
[Acíclico,Cíclico]		Normalmente não utilizado.
[Cíclico,Nulo]		Não em uso comum no momento. Este é um A-Team.
[Cíclico,Acíclico]		Um Blackboard - a estrutura mais popular depois de [Acíclico,Acíclico]
[Cíclico,Cíclico]		Não utilizado.

Figura 2.3: Uma taxonomia de design.

2.3 O Que é um Time Assíncrono ?

2.3.1 Definição

Talukdar e de Souza [TALU93a] definem um Time Assíncrono (Time-A), como qualquer super-agente cujos agentes são autônomos, cujas comunicações entre os mesmos são assíncronas e cujo fluxo de dados é cíclico.

2.3.2 Características Principais

São três os principais aspectos que caracterizam um Time-A, provenientes da própria definição [TALU92b]:

- agentes autônomos : um agente é dito autônomo, se ele próprio decide sobre sua seleção de entrada, planejamento e política de alocação de recursos. Uma vez que os agentes autônomos são completamente independentes uns dos outros, novos agentes podem ser adicionados ao time ou agentes já existentes podem ser retirados sem qualquer aviso e sem qualquer alteração no trabalho dos outros agentes e consequentemente do time.
- comunicações assíncronas : agentes podem ler e escrever informações em memórias compartilhadas sem qualquer sincronização entre si. Isso, aliado à autonomia dos agentes, permite que os mesmos trabalhem o tempo todo em paralelo.
- fluxo de dados cíclico : agentes recuperam, modificam e armazenam informações continuamente nas memórias compartilhadas, de forma que um agente pode alterar resultados gerados por outro agente.

Em um Time-A, cada algoritmo procura apresentar suas melhores soluções que podem ser reutilizadas como entrada para outro algoritmo no time. Um Time-A possui uma característica inerente de cooperação entre os algoritmos, incrementando as chances de se gerar melhores soluções do que cada algoritmo pode fazer sozinho. Devido à não existência de hierarquia nos Times-A, não há necessidade de processos gerenciadores controlarem como os algoritmos cooperam entre si.

2.3.3 Um Exemplo Simples

Um Time-A é formado por agentes que comunicam entre si através de memórias compartilhadas. Graficamente, podemos representar agentes como setas e memórias compartilhadas como retângulos. Os agentes poder ler e escrever nas memórias. Estas por sua vez podem ser compostas de muitas outras memórias. A Fig. 2.4 apresenta um exemplo de Time-A. Neste exemplo,

o agente A lê da memória 3 e escreve na memória 1. O agente B lê da memória 3 e escreve na memória 2. O agente C lê da memória 1 e memória 2 e escreve na memória 3. Todos os agentes são autônomos e assíncronos, o que significa que eles podem operar simultâneamente e continuamente sobre diferentes dados sem sincronização entre si. A condição de fluxo de dados cíclico também é satisfeita, uma vez que há dois ciclos neste fluxo de dados: um composto pela memória 1, agente C, memória 3 e agente A e o outro composto pela memória 2, agente C, memória 3 e agente B.

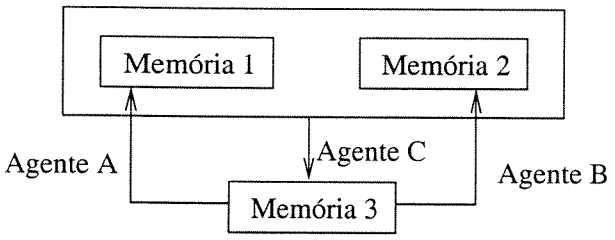


Figura 2.4: Um exemplo de Time Assíncrono.

2.4 Agentes Destruidores

Quando se fala em agentes autônomos, e no modo como trabalham livre e cooperativamente, surge a dúvida : essa livre iteração entre agentes autônomos não faria com que o Time-A gerasse soluções inconsistentes e inúteis?

Para evitar esse problema, usa-se o que se chama de Agentes Destruidores (*Destroyers*) [TALU92b] para apagar ou destruir soluções ruins que não devem mais ser utilizadas. Os agentes destruidores também garantem que o tamanho da memória não cresça indefinidamente uma vez que os algoritmos estão constantemente gerando novas soluções.

Apesar dos destruidores estarem relacionados a eliminação de dados e soluções das memórias, eles são benéficos para a melhora das soluções que o Time-A como um todo pode gerar.

Os agentes destruidores são caracterizados pela sua política de destruição. Esta política define como novos dados são incorporados numa dada memória, ou então que dados serão eliminados da mesma.

2.5 Parâmetros de Design

O projeto de um Time-A é regido por um conjunto de parâmetros essenciais, que se confundem com suas próprias características fundamentais. A seguir serão relacionados os parâmetros utilizados por De Souza [DSOU93] na construção de Times-A para a resolução do TSP. É claro que estes parâmetros podem ser adaptados a outros tipos de problema, que é o caso deste trabalho.

2.5.1 Algoritmos *Consensus-seeking*

Denominam-se algoritmos *consensus-seeking* aqueles que usam duas ou mais soluções para gerar uma nova incorporando informações de todas as soluções de entrada. Há pelo menos duas classes de algoritmos *consensus-seeking*: os algoritmos *common-agreement* e os *suggestion-based.*

Algoritmos *Common-agreement*

Os algoritmos *common-agreement* extraem somente o que é comum entre muitas soluções, e reconstróem as partes que faltam para uma nova solução.

Algoritmos *Suggestion-based*

Ao contrário dos *common-agreement*, os algoritmos *suggestion-based* constróem uma nova solução usando tantas fronteiras quanto possível das soluções originais, ao invés de só as partes comuns como uma configuração inicial para uma nova solução.

2.5.2 Fluxo de Dados

O fluxo de dados é um importante parâmetro de design para um Time-A e normalmente está relacionado a quais algoritmos estão disponíveis. A princípio, deve-se tirar proveito de qualquer informação conhecida a respeito dos algoritmos a fim de se obter o melhor em performance e qualidade das soluções que um Time-A pode fornecer. As Figs. 2.5, 2.6 e 2.7 mostram três configurações com estruturas de fluxo de dados diferentes [DSOU93], de acordo com os algoritmos utilizados por De Souza na resolução do TSP em seu trabalho.

Nas configurações apresentadas pode-se observar mais de um tipo de memória e de agente (algoritmo). A memória "Rotas Completas" contém uma população de soluções para um TSP. Os *Algoritmos Deconstrutores* (não confundir com os agentes destruidores) lêem duas ou mais rotas em "Rotas Completas" e extraem as fronteiras comuns dessas rotas para gerar uma rota fechada infactível que é armazenada em "Rotas Parciais", incluindo fronteiras extras caso necessário. Os *Algoritmos de Construção*, ao contrário, tomam rotas incompletas em "Rotas Parciais" para formar uma solução factível que é armazenada de volta em "Rotas Completas". Na fig 2.6

nota-se a presença de três algoritmos *Suggestion-based* : "Mixer" (MI), "Tree Mixer" (TM) e "Held-Karp" (HK), responsável pela existência da memória "1-tree". Os Algoritmos de Melhoramentos por sua vez, lêem uma rota em "Rotas Completas", tentam alguma modificação na mesma e a reescrevem de novo em "Rotas Completas".
Na última figura, a memória compartilhada "Rotas Completas" passa a ter um novo nome, "Rotas Melhoradas", devido à adição de mais uma memória, "Coarse Tours", que também possui rotas completas, porém são um passo intermediário antes destas serem armazenadas em "Rotas Completas". O agente *Destruidor*, comum a todas as configurações, é responsável direto pela "limpeza" das memórias "Rotas Completas" ou "Rotas Melhoradas".

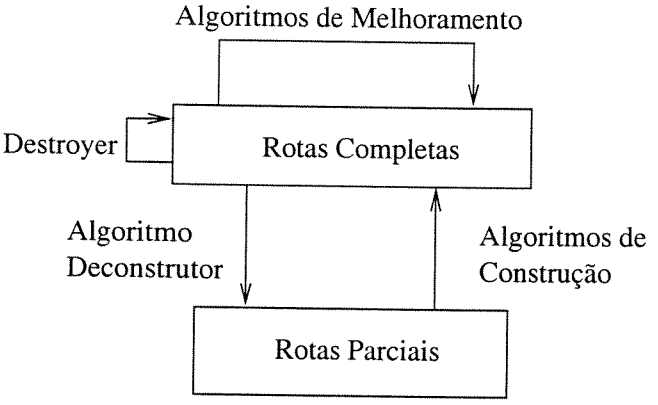


Figura 2.5: Um fluxo de dados de um Time Assíncrono para a resolução de um TSP (primeira configuração).

2.5.3 Estratégia de Inicialização

Antes de todos os algoritmos do Time-A entrarem em funcionamento, faz-se necessária a inicialização das "Rotas Melhoradas" com um certo número de rotas. Os testes realizados por De Souza comprovaram que a memória deve ser inicializada com poucas rotas (menos de 10% do seu tamanho). Para 50% ou mesmo toda a memória, verificou-se que é completamente inútil, uma vez que as piores soluções são substituídas imediatamente após a inicialização, sem qualquer chance de serem selecionadas por qualquer algoritmo.

2.5.4 Política de Seleção

Política de seleção corresponde ao modo como cada algoritmo vai escolher elementos entre os muitos disponíveis nas memórias compartilhadas. De Souza [DSOU93] utilizou três políticas de seleção em seu trabalho : *Head-Tail*, onde o algoritmo correspondente seleciona a rota com o

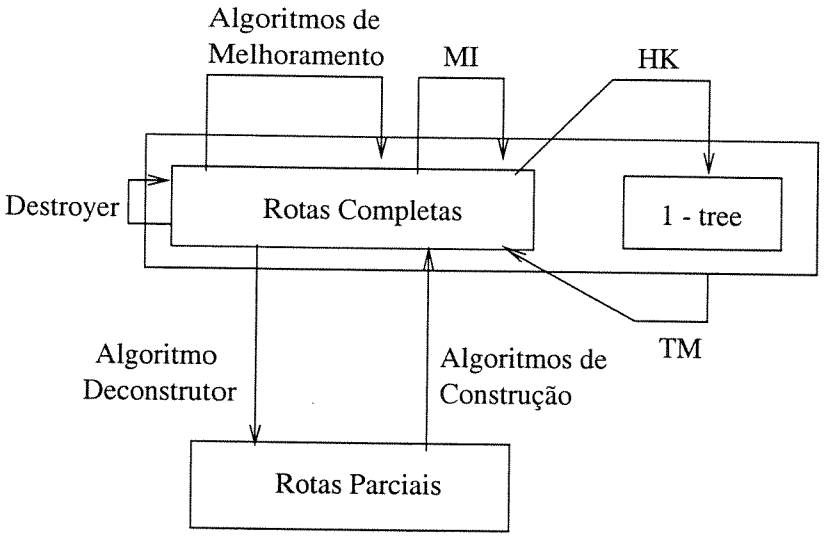


Figura 2.6: Um fluxo de dados de um Time Assíncrono para a resolução de um TSP (segunda configuração).

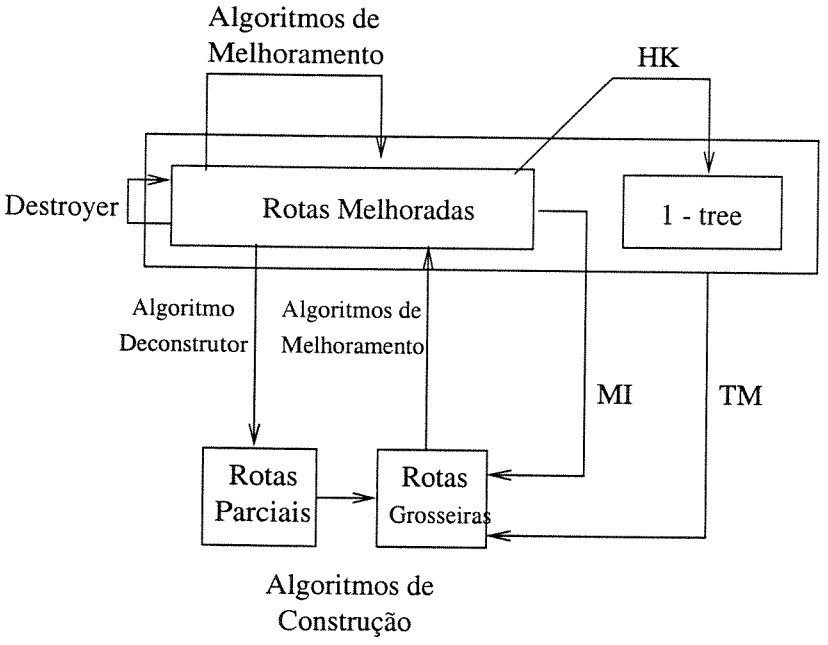


Figura 2.7: Um fluxo de dados de um Time Assíncrono para a resolução de um TSP (terceira configuração).

menor comprimento (*head*) na memória, tal que esta rota não tenha sido selecionada ainda ou gerada por este mesmo algoritmo; *Tail-Head*, que é análoga à anterior mas selecionando rotas com o maior comprimento na memória; e a *Random*, que, obviamente, seleciona aleatoriamente um elemento dentre todos disponíveis na memória.

2.5.5 Política de Destruição

A política de destruição define a estratégia que os destruidores usam para substituir e apagar dados das memórias. Estruturas de memória simples não necessitam de agentes destruidores para controlar suas políticas de destruição; os outros algoritmos podem fazê-lo por si mesmos. É o caso das "Rotas Parciais" e "1-Tree".

Exemplos de políticas de destruição [DSOU93]: substituição da pior rota, substituição de qualquer rota com distribuição de probabilidade uniforme, substituição de qualquer rota com distribuição de probabilidade linear.

2.5.6 Tamanho da Memória

Os tamanhos das memórias são fatores que podem influenciar a performance de um Time Assíncrono ou não. É o caso do Time-A da Fig. 2.7, onde De Souza testou diferentes tamanhos de memória para as "Rotas Melhoradas". Os tamanhos das "Rotas Parciais" e "1-Tree" mostraram quase nenhuma influência no comportamento do time.

2.6 Organizações *Scale Efficient*

Segundo Talukdar [TALU92b], um super-agente que é aberto, isto é, pode crescer facilmente, e cujos membros, novos e velhos, cooperam efetivamente, ou seja, o que justifica o crescimento, pode ser chamado *scale efficient*. No dia-a-dia tem-se vários exemplos desse tipo de organização como por exemplo, determinadas comunidades de peixes que, quanto maiores, menores são as chances de qualquer um de seus membros ser devorado por predadores; certos grupos de pássaros; sociedades de insetos; redes neurais artificiais; redes de computadores.

Em sociedades como das formigas, temos organizações que são abertas e efetivas. A entrada em uma dessas organizações requer apenas as credenciais corretas, como mesmo aspecto, odor, etc. E esses membros dessas mesmas organizações cooperam em tarefas complexas na construção, manutenção e defesa dos seus ninhos e na busca de alimentos.

Em sua tese, De Souza [DSOU93] apresenta vários testes com Times-A diferentes para a resolução do TSP provando a característica *scale efficient* deste tipo de programação.

2.7 Processamento Distribuído e Times Assíncronos

Tratando-se de Times-A, usa-se normalmente algoritmos com granularidade grossa (tempo de comunicação pequeno em relação ao tempo de processamento), o que torna os Times-A adequados para processamento distribuído em uma rede de computadores, onde o mecanismo de comunicação é lento, se comparado, por exemplo, com um computador multiprocessado de alta velocidade. Entretanto, uma rede de computadores distribuídos fornece muito mais flexibilidade em termos de arquitetura que um simples computador, uma vez que em uma rede todos os computadores podem se comunicar uns com os outros de maneira transparente para o usuário. Tal flexibilidade associada com a execução assíncrona de algoritmos, dá aos Times-A o alto potencial de exploração de paralelismo.

Fato é : quanto maior o tamanho do problema a ser resolvido por um time, maior o tempo total de processamento. Uma vez que o tempo de processamento cresce mais rapidamente que o tempo de comunicação, o custo de comunicação relativo com respeito ao número de processadores diminui, o que também torna a execução em paralelo mais fácil, assim que o problema cresce.

Uma característica importante : devido a não existência de sincronismo entre os agentes de um Time-A, quando eles são designados a vários processadores eles os mantém ocupados todo o tempo fazendo algum trabalho útil. Daí então, espera-se que a eficiência (relação entre *speedup* e número de processadores) de um Time Assíncrono seja muito próxima a um.

Uma outra consequência importante das iterações assíncronas é a robustez. Caso um algoritmo ou processador caia fora de um time, haveria uma distorção de seu fluxo de dados, mas todos os outros algoritmos continuariam trabalhando e, esperançosamente, boas soluções seriam alcançadas. Se há múltiplas cópias de um mesmo algoritmo que morreu em diferentes processadores, então não ocorreria nenhuma distorção no fluxo de dados, e sim apenas uma distorção na frequência relativa de iteração entre os algoritmos. Isto implica que, maior o número de ciclos em um fluxo de dados de um Time-A e maior o número de processadores para rodar um Time-A, mais confiável esse time é.

Capítulo 3

Métodos Heurísticos Construtivos

3.1 Introdução

Neste capítulo serão apresentados de forma resumida os métodos de planejamento da expansão da transmissão utilizados nos agentes do Inicializador. Estes correspondem aos Métodos Heurísticos Construtivos, também chamados de "Métodos Aproximados", pois seus resultados são apenas valores aproximados, não apresentando a mesma precisão que algoritmos combinatoriais como Algoritmos Genéticos ou Busca Tabu. Porém, os Métodos Construtivos ainda são utilizados em concessionárias de energia elétrica, pois foram introduzidos em uma época cujos recursos computacionais não eram tão desenvolvidos e de fácil acesso como nos dias de hoje.
Neste trabalho, serão apresentados o Método de Garver em três versões variantes, o Método do Mínimo Esforço em duas versões variantes e o Método do Mínimo Corte de Carga em apenas uma versão. Isto resulta, no final, em seis agentes Construtores que possibilitam diversas estruturas do Time-A desenvolvido neste trabalho.

3.2 Método de Garver

3.2.1 Introdução

O modelo formulado por Garver [GARV70], foi a primeira proposta para planejamento de redes de transmissão baseado em um modelo de transportes (programação linear). Esta metodologia consiste basicamente em resolver de maneira aproximada uma versão relaxada do modelo DC. Neste modelo somente se leva em conta a Primeira Lei de Kirchhoff e a capacidade de transmissão das linhas, sendo a Segunda Lei de Kirchhoff completamente desprezada. É um processo chamado de passo-a-passo onde em cada iteração do algoritmo se toma uma decisão de

adicionar uma linha à configuração atual. A linha que é adicionada é aquela que aparece mais sobrecarregada quando o modelo de transportes é resolvido.

Na metodologia de Garver, todo fluxo que não puder ser transportado pelas ligações normais fluirá pelas ligações de sobrecarga, pois estas têm capacidades ilimitadas, e só passará através das ligações de sobrecarga quando for impossível transportá-lo pelas ligações normais, pois estas últimas têm custos muito inferiores. Em cada estágio do processo de planejamento, deve-se resolver um problema de programação linear e logo adicionar um circuito na trajetória de maior sobrecarga. O processo é repetido até eliminar todas as sobrecargas.

A vantagem da metodologia de Garver é a simplicidade na implementação do algoritmo pois ela exige somente soluções sucessivas de programação linear. As maiores limitações são: relaxamento da Segunda Lei de Kirchhoff; obtenção de soluções sub-ótimas. Assim, apesar de ser baseada em um método exato de solução de problemas de otimização (por exemplo, o método Simplex), trata-se de fato de uma metodologia heurística.

Supondo conhecidos os valores de geração e de carga de um dado sistema, tem-se os seguintes passos na estimação da rede de transmissão [GARV70] :

1. Formular as equações de fluxo de potência como um problema de minimização linear.
2. Usar programação linear para resolver o problema de minimização para as alterações na rede necessárias. Este resultado é chamado de estimação de fluxo linear.
3. Selecionar uma adição de linha baseada na localização da maior sobrecarga nesta estimativa de fluxo.
4. Repetir os passos da estimação de fluxo e da seleção de linha até que não haja qualquer sobrecarga.

A mais importante inovação da técnica de estimação de fluxo foi o fato de que sobrecargas não irão aparecer nas linhas mas em um novo tipo de ligação de rede chamada ligação de sobrecarga ou caminho de sobrecarga. Ligações de sobrecarga existem entre todas as barras na rede em questão. Elas inicialmente fornecem rotas para sobrecargas, quando elas ocorrem, e mais tarde locais para novas linhas. Sendo assim, o método considera então dois tipos de ligações:

- **Ligações Normais**: Com capacidade de transmissão máximas iguais às linhas reais e custos de transporte iguais às reatâncias dessas linhas.
- **Ligações de Sobrecarga**: São ligações fictícias com capacidades de transmissão ilimitadas e custos de transporte muito superiores aos das linhas normais; estas ligações são colocadas entre todos os nós ou barras nas quais sejam permitidas a construção de novas linhas.

A metodologia foi usada em três versões (agentes): *GarverA*, *GarverB* e *GarverC*.

Os algoritmos utilizados [GALL97] são descritos a seguir:

3.2.2 GarverA

(1) Resolve-se um Fluxo de Carga DC do sistema, utilizando-se aqui o pacote MINOS [MURT87] para os cálculos envolvendo programação linear. O banco de dados da rede já considera a presença das barras e linhas fictícias (ligações de sobrecarga).

(2) Com os resultados da análise DC anterior, calcula-se o fluxo de potência para cada ramo do sistema.

(3) Ordena-se os ramos pelo valor dos respectivos fluxos de sobrecarga, em ordem decrescente.

(4) Se o fluxo do ramo mais sobrecarregado for menor que um limite preestabelecido no início do programa, então vá para **(6)**. Caso contrário, o programa executa a adição de novas linhas à rede, o que pode ser feito de duas maneiras, dependendo dos parâmetros definidos nos arquivos de configuração, os quais variam de acordo com o sistema elétrico considerado. Na primeira, pode-se simplesmente adicionar uma linha ao ramo mais sobrecarregado. Na segunda, é feita uma análise da necessidade da criação de um caminho completo, além da adição de uma simples linha ao mesmo ramo mais sobrecarregado, a fim de evitar a presença de elementos desconexos no sistema.

(5) Voltar para **(1)**.

(6) Efetua-se o *refinamento* do sistema, que consiste na verificação da possibilidade de se remover linhas adicionadas desnecessariamente.

3.2.3 GarverB

O algoritmo *GarverB* consiste em uma variante do *GarverA* descrito acima. A diferença está em que a ordenação dos ramos para a seleção é realizada baseada no número de linhas requeridas para adição em cada ramo, ao invés do fluxo máximo de sobrecarga.

3.2.4 GarverC

O algoritmo *GarverC* apresenta a mesma estrutura que o algoritmo *GarverB*, exceto pela introdução dos seguintes passos, previamente ao inicio do processo de estimação:

- correção de caminhos conectando-se linhas eletricamente isoladas, ou seja, linhas onde não há um fluxo de potência através das mesmas;

- correção de caminhos entre barras fictícias, para que a capacidade de transmissão das linhas em série seja suficiente de acordo com o requisitado;
- adição de linhas cuja fração seja superior a um valor de entrada.

3.3 Método de Mínimo Esforço

3.3.1 Introdução

Esta metodologia [MONT82] foi desenvolvida na Unicamp (em parceria com o CEPEL e a Eletrobrás), onde os pesquisadores desenvolveram um software interativo para o planejamento da expansão da transmissão a longo prazo. Neste trabalho toda a análise de rede é baseada no modelo de fluxo de carga DC. Aqui também se faz um plano de expansão passo-a-passo, isto é, para uma configuração da rede os circuitos são adicionados um a um ou em pequenos grupos. A ordenação de novas adições é basedada num critério denominado de "mínimo esforço", que leva em conta o padrão de distribuição de fluxo na rede.

A parte do software referente à análise de rede automática é dividida em duas fases:
a Fase-1 reforça automaticamente a rede até que todas as sobrecargas sejam eliminadas e todas as barras desconectadas con injeções não-nulas sejam conectadas. O critério de reforço ou adição é baseado na análise custo/benefício dos "caminhos de mínimo esforço" na rede em questão. Esses caminhos são determinados com a ajuda do critério de mínimo esforço. Na Fase-2 a rede final é analisada levando-se em conta o efeito de simples contingências: o algoritmo inicialmente simula o efeito da remoção de elementos do circuito selecionados de uma lista de contingências mais severas. As sobrecargas resultantes são usadas para identificar uma barra "crítica". Daí então aplica-se novamente a Fase-1 a todos caminhos de mínimo esforço com origem nesta barra. O procedimento é repetido até que a rede satisfaça a restrição de segurança estática.

O problema de desconexões na rede inicial é contornado adotando-se, superposta à configuração do sistema, uma "rede fictícia" constituída por ligações com reatâncias muito maiores que o normal (10^4 por exemplo). Estas ligações são colocadas em todos os ramos onde novas adições são possíveis. Os ramos da rede fictícia somente serão utilizados quando o transporte de potência não for factível na rede real, ou seja, em todos as situações nas quais há fluxo de potência entre áreas isoladas.

A Fase-1 do algoritmo apresentado em [MONT82], que é a utilizada como base para os agentes desenvolvidos nesta tese, é descrita como segue:

(1) Resolva o fluxo de carga DC. Se não houver qualquer sobrecarga ou desconexão, vá para **(7)**.

(2) Calcule o índice de mínimo esforço Δz_i para cada possível adição (a descrição do índice de

mínimo esforço é detalhada logo adiante).

(3) Agrupe as possíveis adições de linhas e transformadores de acordo com suas respectivas classes de tensão.

(4) Selecione os n melhores candidatos de cada classe ordenados de acordo com o índice Δz_i/custo. Eles constituirão os primeiros ramos dos caminhos de mínimo esforço.

(5) Se ambos os nós terminais de cada circuito selecionado não são nós intermediários (nós isolados com injeções de rede nulas), o caminho tem apenas um elemento; vá para o passo **(6)**. Caso contrário, determine as possíveis adições conectadas a cada nó intermediário e adicione ao caminho o circuito com a maior relação Δz_i/custo; repita o procedimento até que ambos os nós terminais não sejam de tipo intermediário.

(6) Simule a adição de cada caminho de mínimo esforço ao sistema e avalie a redução de sobrecarga (benefício). Adicione os custos dos circuitos para obter o custo do caminho. Adicione ao sistema o caminho de mínimo esforço com a menor relação custo/benefício; vá para o passo **(1)**.

(7) Alguns dos circuitos adicionados podem ter um pequeno fator de carregamento na rede final; ordene-os decrescentemente pela razão custo/fluxo-MW e elimine aqueles cuja remoção não cause sobrecargas ou desconexões.

3.3.2 Ordenando as adições

A solução do problema:

$$\begin{aligned} Min \quad & z = \frac{1}{2}\sum_{j=1}^{M} x_j T_j^2 \\ s.a. \quad & \sum_{j\in\Omega_k} T_j = P_k; k = 0, N \end{aligned} \tag{3.1}$$

onde:

M Número de ramos;

$N+1$ Número de barras;

j Ramo conectando as barras k e l;

Ω_k Conjunto de ramos conectados à barra k.

é idêntica à solução do fluxo de carga linearizado, isto é, a distribuição dos fluxos numa rede segue uma lei de mínimo esforço que minimiza o produto das reatâncias *p.u.* de cada ramo pelo quadrado do respectivo fluxo. Esta função de mínimo esforço é utilizada aqui como um índice de desempenho para ordenar as adições mais atrativas; a adição que afeta z o máximo é a melhor favorável do ponto de vista da distribuição de fluxo "natural". A sensibilidade do índice de performance z com respeito à susceptância do ramo i é dada por:

Esta análise baseia-se no fato de que a distribuição dos fluxos em uma rede segue uma *lei de mínimo esforço* que minimiza o produto das reatâncias *p.u.* de cada ramo pelo quadrado do respectivo fluxo. Esta função de mínimo esforço é utilizada aqui como um índice de desempenho para ordenar as adições mais atrativas, da seguinte maneira:

$$\frac{\partial z}{\partial \gamma_i} = -\frac{1}{2}\psi_i^2 \tag{3.2}$$

onde ψ_i é o ângulo através do ramo i antes das adições. A variação de z devido à adição de uma linha com susceptância $\Delta\gamma_i$ no ramo i é dada aproximadamente por:

$$\Delta z_i = -\frac{1}{2}\psi_i^2 \Delta\gamma_i \tag{3.3}$$

Em cada passo do processo de planejamento é adicionado ao sistema aquele circuito que produz o maior impacto na distribuição de fluxos na rede, isto é, aquele que apresenta o maior valor de Δz_i.

Este critério do mínimo esforço foi utilizado em duas versões de agentes implementados: *minD* e *minE*.

3.3.3 minD

O programa *minD* possui algoritmo semelhante aos anteriores com diferenças em alguns itens, devidas ao novo critério:

(1) Resolve-se um Fluxo de Carga DC do sistema, utilizando-se aqui o pacote MINOS [MURT87] para os cálculos envolvendo programação linear. O banco de dados da rede já considera a presença das barras e linhas fictícias (ligações de sobrecarga).

(2) Com os resultados da análise DC anterior, calcula-se o somatório das sobrecargas do sistema e o índice de mínimo esforço para cada ramo do sistema.

(3) Ordena-se os ramos pelo valor dos respectivos índices de mínimo esforço.

(4) Se o somatório das sobrecargas do sistema for menor que um limite preestabelecido no início do programa, então vá para (6). Caso contrário, o programa executa a adição de novas linhas à rede, o que pode ser feito de duas maneiras, assim como em GarverA, dependendo dos parâmetros definidos nos arquivos de configuração, os quais variam de acordo com o sistema elétrico considerado. Na primeira, escolhe-se aleatoriamente um dos três primeiros ramos da lista ordenada para a adição de uma linha ou de um caminho. Na segunda, é feita uma análise da necessidade da criação de um caminho completo, além da adição da linha escolhida aleatoriamente ao mesmo ramo mais sobrecarregado, a fim de evitar a presença de elementos desconexos no sistema.

(5) Voltar para (1).

(6) Efetua-se o *refinamento* do sistema, que consiste na verificação da possibilidade de se remover linhas adicionadas desnecessariamente.

3.3.4 minE

A única diferença deste programa em relação ao *minD* consiste na introdução de uma normalização pelo custo de cada ramo, quando do cálculo do índice de mínimo esforço, ou seja, o valor calculado do índice para cada ramo é dividido pelo custo correspondente.

3.4 Método de Mínimo Corte de Carga

3.4.1 Introdução

A maioria dos índices de performance imediatos em sistemas de transmissão são relacionados a sobrecargas de linhas ou violações de limites de tensão em barras. Muitos algoritmos de ordenação foram desenvolvidos de acordo com essas medidas. Mas uma limitação na aplicação destes índices relacionados a sobrecarga ou tensão é que eles são apenas medidas indiretas de adequação do sistema, que é fundamentalmente medida por corte de carga. Consequentemente, estes índices não são compatíveis com índices usados nos estudos de planejamento da geração, que são diretamente relacionados a perda de carga no sistema. Com isto em mente, pesquisadores investigaram novos índices de performance para os sistemas de geração/transmissão que pudessem ser traduzidos em termos de corte e suprimento de carga.

De maneira semelhante ao Método de Mínimo Esforço, o Método do Mínimo Corte de Carga [PERE85a] realiza a adição de linhas selecionadas de acordo com um índice de sensibilidade

que permite encontrar linhas mais atrativas. Este algoritmo também é chamado de algoritmo passo-a-passo, pois em cada iteração deve-se decidir a adição de um circuito na configuração base até que o sistema opere adequadamente, isto é, sem corte de carga.

3.4.2 Modelamento da Rede

Quando as resistências série e as admitâncias shunt dos ramos são ignoradas, a distribuição do fluxo de potência ativa numa rede composta de N barras pode ser aproximadamente calculada pelo fluxo de carga DC.

$$B\theta + g = d \tag{3.4}$$

onde:

g é o vetor de geração (Nx1) ;

d é o vetor de carga (Nx1) ;

θ é o vetor dos ângulos das tensões nodais (Nx1) ;

B é a matriz susceptância (NxN).

O sistema linear 3.4 é resolvido por técnicas de esparsidade. O vetor solução Θ pode ser utilizado para estimar os fluxos da rede. A potência ativa f_{kl} é dada por:

$$f_{kl} = (\theta_k - \theta_l)\gamma_{kl} \tag{3.5}$$

para todo $k = 1, 2, ..., N$
para todo $l \in \Omega_k$

3.4.3 O Índice de Mínimo Corte de Carga

Se o fluxo de potência f_{kl} estimado a partir da solução do sistema linear 3.4 excede seu limite $\overline{f}_{kl}$, o sistema estará sobrecarregado. A eliminação de sobrecargas envolve o redespacho da geração g e, se necessário, medidas mais drásticas como corte de carga em algumas ou mesmo todas as barras de carga.

O problema de mínimo corte de carga consiste em minimizar o total de corte de carga no sistema. Pode ser calculado como a solução do seguinte problema de programação linear:

$$\begin{array}{lll} Min & z = \sum_{k=1}^{N} r_k & Variaveis\ Duais \\ s.a. & & \end{array} \tag{3.6}$$

$$\begin{array}{ll} B\theta + g + r = d & \pi_d \\ 0 \geq -g \geq -\overline{g} & \pi g \\ -r \geq -d & \pi r \\ -|S\theta| \geq -\overline{\psi} & \pi\psi \end{array} \tag{3.7}$$

onde:

r é um vetor de gerações fictícias que corresponde ao corte de carga em cada barra;

S é a matriz incidência nó-ramo;

$\overline{\psi}$ é o máximo ângulo através do ramo $k-l$;

πd, πg, πr e $\pi\psi$ são os vetores de multiplicadores Simplex associados à cada restrição do problema.

Há dois modos de se identificar linhas "críticas" em uma rede de transmissão:

1. sensibilidade com respeito ao limite de fluxo $\overline{f}$
2. sensibilidade com respeito à susceptância de linha γ

Neste trabalho foi considerada a análise de sensibilidade com respeito à susceptância das linhas, que pode ser calculada facilmente pela solução ótima do problema de PL. O índice de desempenho utilizado para a adição dos circuitos é dado pela expressão [PERE85a]:

$$\pi_\gamma(k,l) = (\pi_d(k) - \pi_d(l))(\theta_l^* - \theta_k^*) \tag{3.8}$$

onde:

$\pi_\gamma(k,l)$ é o índice de sensibilidade com respeito a variação na susceptância γ_{kl};

θ_k^* e θ_l^* são os ângulos de tensão da solução ótima quando da análise DC do sistema.

O índice de sensibilidade é um indicador do impacto que produziria a adição de um circuito no corte de carga de um sistema se o circuito fosse adicionado ao sistema elétrico. Assim, aquele circuito que possui o maior valor absoluto do índice de sensibilidade deve ser adicionado à configuração base pois é o melhor candidato para produzir uma maior dimiuição no corte de carga do sistema.

Neste método elaborou-se uma única versão de agente: *mccF*.

3.4.4 mccF

O programa *mccF* também mantém a mesma estrutura dos anteriores apenas com diferenças associadas ao novo índice:

(1) Resolve-se um Fluxo de Carga DC do sistema, utilizando-se aqui o pacote MINOS [MURT87] para os cálculos envolvendo programação linear. O banco de dados da rede já considera a presença das barras e linhas fictícias (ligações de sobrecarga).

(2) Calcula-se o corte de carga do sistema (via MINOS), e o índice de mínimo corte de carga, aqui já normalizado pelo custo, para cada ramo do sistema.

(3) Ordena-se os ramos pelo valor dos respectivos índices de corte de carga.

(4) Se o corte de carga do sistema for menor que um limite preestabelecido no início do programa, então vá para **(6)**. Caso contrário, o programa executa a adição de novas linhas à rede, o que pode ser feito de duas maneiras, assim como em GarverA, dependendo dos parâmetros definidos nos arquivos de configuração, os quais variam de acordo com o sistema elétrico considerado. Na primeira, escolhe-se aleatoriamente um dos três primeiros ramos da lista ordenada para a adição de uma linha ou de um caminho. Na segunda, é feita uma análise da necessidade da criação de um caminho completo, além da adição da linha escolhida aleatoriamente ao mesmo ramo mais sobrecarregado, a fim de evitar a presença de elementos desconexos no sistema.

(5) Voltar para **(1)**.

(6) Efetua-se o *refinamento* do sistema, que consiste na verificação da possibilidade de se remover linhas adicionadas desnecessariamente.

Capítulo 4

Inicializador Proposto

4.1 Introdução

Neste capítulo são detalhadas as características de programação e funcionamento dos componentes do Time-A proposto para a inicialização do problema do planejamento da transmissão. Falando de uma maneira geral, o Inicializador (como será chamado daqui em diante o Time Assíncrono para a Inicialização do Problema do Planejamento da Expansão) possui a estrutura mais simples possível para um Time-A, levando-se em conta as várias configurações propostas por de Souza [DSOU93]. É composto por apenas uma memória compartilhada denominada *Memória Central*, pelo agente *Destruidor* e pelos Agentes Construtores que contém os métodos heurísticos construtivos descritos no capítulo anterior. Estes componentes, processos UNIX implementados em FORTRAN77, trabalham em paralelo através do pacote *PVM*, distribuídos em uma rede homogênea de *workstations SUN-SPARC4*.

4.2 Estrutura do Inicializador

O Inicializador é composto de três componentes principais:

- **Memória Central** : é a única memória compartilhada do sistema, na qual os agentes atuam lendo e escrevendo dados;
- **Agentes Construtores** : lêem dados na Memória Central, alteram esses dados e os reescrevem de volta na mesma, não havendo qualquer comunicação entre si;
- **Agente Destruidor** : responsável por manter constante o número de elementos armazenados na Memória Central.

A disposição destes componentes dentro desta estrutura pode ser visualizada no exemplo da Fig.4.1. As linhas tracejadas representam os processos que exercem as funções dos correspondentes componentes envolvidos pelas mesmas. Os seguintes processos (programas) foram implementados podendo-se montar diversas configurações para o Inicializador :

- *inic* : programa principal, que exerce principalmente as funções da Memória Central e de agente Destruidor;
- *GarverA* : programa correspondente ao agente construtor GarverA, que utiliza a metodologia de Garver baseada no fluxo;
- *GarverB* : programa correspondente ao agente construtor GarverB, que utiliza a metodologia de Garver baseada no número de linhas requeridas;
- *GarverC* : programa correspondente ao agente construtor GarverC, que possui a estrutura básica de *GarverB* mais algumas correções específicas;
- *minD* : programa correspondente ao agente construtor minD, que utiliza o método do Mínimo Esforço;
- *minE* : programa correspondente ao agente construtor minE, que utiliza o método do Mínimo Esforço, porém com uma normalização no cálculo do índice;
- *mccF* : programa correspondente ao agente construtor mccF, que utiliza o método do Mínimo Corte de Carga.

Desta forma, pode-se observar no mesmo exemplo da Fig.4.1 uma configuração formada por um agente *GarverA*, um agente *minD* e um agente *mccF*.

4.3 Arquivos que Compõem o Inicializador

Além dos arquivos de programas já mencionados na seção anterior, há um programa auxiliar para a retirada de agentes durante o processamento do time e mais alguns arquivos que devem ser relacionados. Estes arquivos são necessários para o funcionamento do Time-A proposto neste trabalho, uma vez que estes contêm dados dos sistemas elétricos, parâmetros para o planejamento, além de parâmetros relativos ao processamento paralelo do sistema. Eis os arquivos e seus respectivos conteúdos :

- *opcoes*: número de linhas do sistema elétrico que está sendo utilizado; número inicial de agentes do time; especificações de cada agente, isto é, o código do agente e a respectiva máquina onde o mesmo irá rodar;

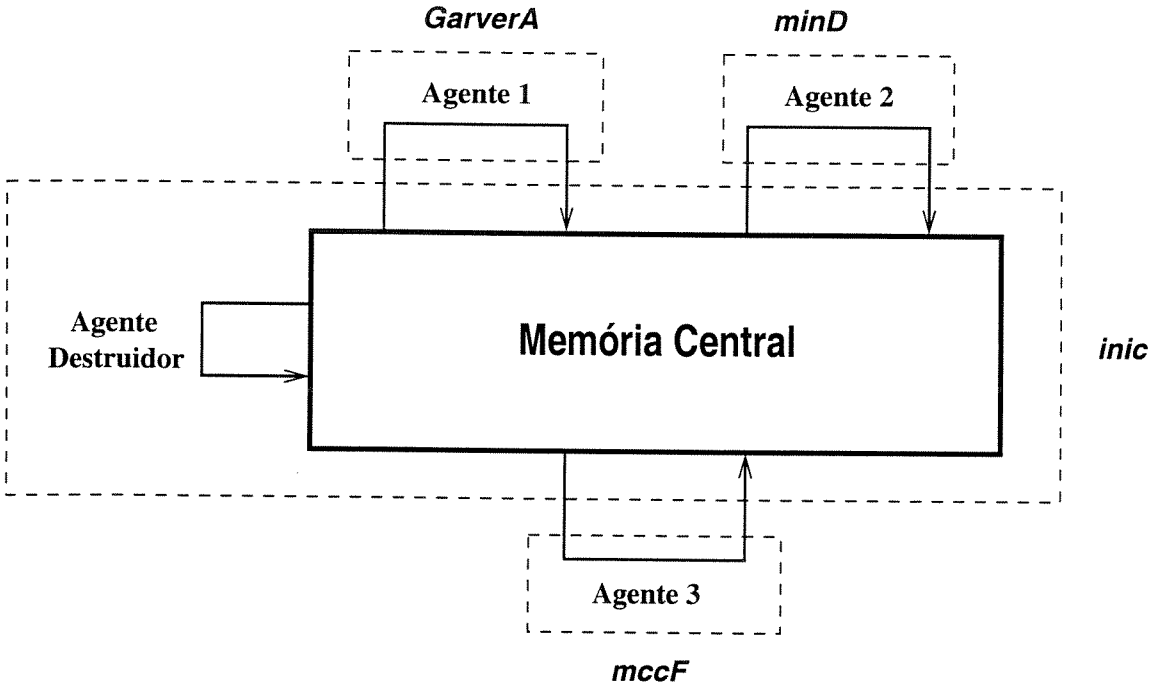


Figura 4.1: Exemplo de Inicializador com três agentes.

- *rede.dat*: dados de barras e ramos do sistema elétrico considerado, já levando em conta valores futuros de geração, carga e capacidade de transmissão;
- *param.dat*: número de configurações iniciais a serem geradas; número aleatório utilizado no funcionamento do Time-A; sobrecarga permitida para fluxos de potência em linhas e trafos; valor que especifica para quais tamanhos de redes serão feitas correções adicionais; número máximo de alternativas de caminhos a serem consideradas no momento da inserção de ramos;
- *paramin.dat*: valores que especificam o modo como a lista ordenada dos ramos a serem adicionados será criada nos agentes minD e minE, com relação à influência da sobrecarga e do índice de sensibilidade (por ex., os três primeiros elementos da lista serão ordenados por sobrecarga e os dois últimos pelo índice);
- *nlin.dat*: número de linhas fictícias conectadas a cada barra do sistema;
- *nconec.dat*: relação das linhas fictícias conectadas a cada barra do sistema;
- *mpop.dat*: saída do sistema, com as configurações iniciais geradas pelo Inicializador;
- *remv*: programa através do qual agentes podem ser removidos do time;
- *seed*: contém número inteiro usado como semente para a geração de números aleatórios necessários ao programa *inic*.

4.4 Estrutura e Funcionamento dos Programas

4.4.1 Uma Visão Geral

O Inicializador apresentado neste trabalho tem como objetivo gerar um número determinado de configurações iniciais para o planejamento da expansão do sistema elétrico contido em *rede.dat*, que será efetuado por métodos mais "pesados" como Algoritmos Genéticos, *Simulated Annealing* e Busca Tabu. Os valores de *rede.dat* já levam em conta valores futuros de geração, carga e capacidade de transmissão.

Antes de qualquer coisa, deve-se configurar a Máquina Virtual Paralela (via *PVM* [GEIS94]), isto é, inicializar um *daemon PVM* em cada estação de trabalho da rede onde se pretende executar um ou mais agentes (levando em conta também a estação onde irá r o processo central *inic*).
Isto feito, e considerando que todos os arquivos de parâmetros do Inicializador estejam configurados corretamente, deve-se executar o programa *inic* em uma das *workstations* da Máquina Virtual Paralela. Este, por sua vez, se encarregará de desovar os agentes iniciais do Time-A nas outras *workstations*, que também fazem parte da MVP.
Uma vez desovados os agentes contidos em *opcoes*, o processo *inic* passa a executar a função

de Memória Central, onde irá armazenar as n configurações que serão geradas a partir do zero (valor de n lido em *param.dat*). Sendo assim, *inic* irá simular uma memória compartilhada (na ausência de um computador de memória compartilhada), controlando o acesso dos agentes aos elementos da população de configurações. Dentro desse modo de controle, está embutido o agente Destruidor, pois cada novo valor calculado pelos agentes é armazenado de volta na mesma posição de memória de *inic*, daí o número de elementos (configurações) da população de soluções permanece constante todo o tempo.
Durante o processamento do Inicializador, agentes podem ser adicionados ao time, simplesmente entrando-se com o nome do programa correspondente diretamente no console de uma das estações da MVP. Agentes também podem ser removidos do time, através do programa *remv*.
A partir de um certo instante do processamento, *inic* terá conhecimento das configurações que estão prontas (maiores detalhes serão discutidos nas subseções a seguir), bloqueando-as de tal forma que não possam ser mais acessadas e consequentemente alteradas por qualquer agente. Quando todas estiverem "definitivamente bloqueadas", o Inicializador encerra seu processamento, retornando as configurações obtidas em *mpop.dat*.

Assim, tem-se uma visão geral do funcionamento dos componentes do Time-A Inicializador. A seguir, esse funcionamento será descrito e analisado em detalhes. No caso dos agentes, será apresentada apenas a estrutura do programa *GarverA*, uma vez que os outros agentes funcionam da mesma maneira, com diferença apenas no algoritmo embutido, que não é o propósito desta discussão.

4.4.2 *INIC*

O programa *inic* pode ser descrito divindo-se o mesmo nas seguintes etapas :

1. Inicialização de variáveis e comandos iniciais do *PVM*.

   Nesta etapa, vale ressaltar a criação do grupo dinâmico [GEIS94] denominado "iniciar". Todos os processos do Time-A farão parte obrigatoriamente deste grupo, para que possam ser identificados pelo seu *rank*, valor que será necessário caso algum agente venha a ser removido por *remv*. A existência do grupo também é importante quando se acrescentam novos agentes, pois estes não terão o *ID* do processo *inic* na máquina virtual, mas este poderá então ser obtido uma vez que *inic* possui *rank* igual a zero necessariamente, possibilitando então a comunicação deste agente com a Memória Central.

2. Leitura do arquivo *seed*, que contém uma semente para a geração de números aleatórios.

3. Leitura dos arquivos *param.dat* e *opcoes*.

   No arquivo *param.dat* é lido o número de configurações a serem geradas pelo Inicializador. No arquivo *opcoes*, são lidos o número de ramos do sistema elétrico considerado e o número de agentes iniciais do time. Este valor pode ser nulo, o que neste caso faz com que *inic* vá para a etapa 5, e que os agentes sejam todos adicionados manualmente.

4. Nova leitura de *opcoes* e desova de agentes.

   Agora, serão lidos os valores que identificam quais são os agentes iniciais e suas respectivas *workstations*. À medida que cada par de valores é lido no arquivo, o processo correspondente é desovado na sua estação e começa imediatamente o seu papel de agente. Após o primeiro agente ter sido desovado, um cronômetro interno é disparado, daí será obtido o tempo total de processamento do Inicializador. Observar que este tempo total não corresponde ao tempo de execução de *inic*, mas sim do início do processamento dos agentes rumo ao seu objetivo, as configurações iniciais.

5. Início do *loop* principal do programa.

   Aqui, *inic* checa o *buffer* e verifica a presença de mensagem(s) proveniente(s) de um ou mais agentes. Essas mensagens são armazenadas no *buffer* segundo uma fila tipo *FIFO*, na ordem em que os processos as enviam para *inic*. Dessa forma, todas informações recebidas são consideradas, uma a uma, por ordem de chegada.

6. Verificação da mensagem recebida.

   Caso haja mensagem no *buffer*, *inic* irá então verificar a natureza da mesma, que pode ser de dois tipos. Num primeiro caso, *inic* estará recebendo uma solicitação de algum agente que acabou de ser adicionado pelo próprio *inic* ou manualmente, para que lhe seja enviado um elemento do vetor *mempop*, que é o lugar onde são armazenadas as configurações. Esta ação corresponde, em termos de Times-A, ao ato do agente ir buscar um elemento da população na Memória Central para começar a trabalhar.
   Num segundo caso, tem-se uma nova configuração, isto é, algum agente retirou um elemento (configuração) da Memória Central anteriormente, fez uma alteração neste elemento e agora está colocando-o de volta na população.

7. Atualização da Memória Central.

   Aqui, a nova configuração enviada pelo agente é armazenada de volta no vetor *mempop*, na mesma posição de onde foi lida antes de alterada.

8. Verificação do término do processamento.

   Neste momento, *inic* verifica se todas as configurações atingiram seu estado final, ou seja, se o Inicializador terminou sua tarefa. Em caso afirmativo, *inic* fecha o cronômetro, salva os resultados em *mpop.dat* e executa comandos finais do *PVM*.

9. Bloqueio definitivo de configuração.

   Juntamente com a nova configuração, *inic* recebe também um sinal que indica se esta mesma configuração atingiu seu estado final, ou seja, se ela foi "terminada". Em caso afirmativo, ela é mantida então definitivamente bloqueada, de tal forma que nenhum agente poderá mais acessar sua correspondente posição na Memória Central e, consequentemente, alterá-la. Caso contrário, ela é liberada e fica armazenada em *mempop* até que seja "retirada" por qualquer um dos agentes ativos no time.

10. Aviso de encerramento para agente.

    Ainda considerando esta nova configuração que chegou em *inic*, supondo-se que a mesma seja definitivamente bloqueada, *inic* então verifica se há alguma posição de *mempop* que esteja disponível, isto é, se existe ainda alguma configuração que não esteja pronta, e daí a envia para o mesmo agente cuja configuração alterada acabou de ser bloqueada em definitivo. Caso não haja mais configurações disponíveis, *inic* envia um sinal para o agente determinando o seu encerramento e a consequente saída do Time-A.

11. Verificação de saída de agente.

    Quando um agente é retirado do time por *remv*, um sinal é enviado por esse mesmo agente para o processo *inic* avisando-o da sua saída. Isso é necessário, pois, quando o agente retira um elemento da população, esta passa a estar num estado de bloqueio, em outras palavras, a posição de memória em *mempop* ocupada por essa configuração fica bloqueada para que nenhum outro processo agente tenha acesso, assim evitando que dois agentes trabalhem sobre os mesmos dados. Daí a necessidade desse sinal, pois sempre o agente estará bloqueando alguma posição, e esta precisa ser desbloqueada para os outros antes que ele seja retirado do time.
    Desta forma, neste ponto do programa *inic*, ele checa a presença deste sinal. Em caso afirmativo, ele retorna para receber outra mensagem, não enviando outro elemento para aquele agente que estaria esperando, mas que na verdade está saindo (isto será melhor explicado adiante em **Observações Importantes**).

12. Escolha de configuração a ser retirada pelo agente.

    Neste instante então, *inic* escolhe aleatoriamente uma entre as n configurações disponíveis no momento (configurações que não estão sendo atualizadas por nenhum agente) e a bloqueia para ser utilizada pelo agente ocioso.

13. Envio da configuração escolhida.

    Em seguida, *inic* envia o elemento que acaba de ser escolhido em *mempop* para o mesmo agente cuja nova configuração recebida acaba de ser rearmazenada, o qual se encontra em estado de espera. Neste instante, esta configuração passa a "pertencer" a este agente, e nenhum outro poderá acessá-la. Daí, *mempop* passa a ter uma configuração disponível a menos.
    Em termos de Times-A, esta situação se passa de maneira inversa, pois o agente é que vai buscar uma nova configuração na Memória Central para ser atualizada, e não o contrário, que é o que acontece em termos de programação.

14. Retorna para a etapa 5.

Na Fig. 4.2 é apresentado um diagrama de blocos onde pode-se visualizar melhor o funcionamento de *inic*, descrito acima.

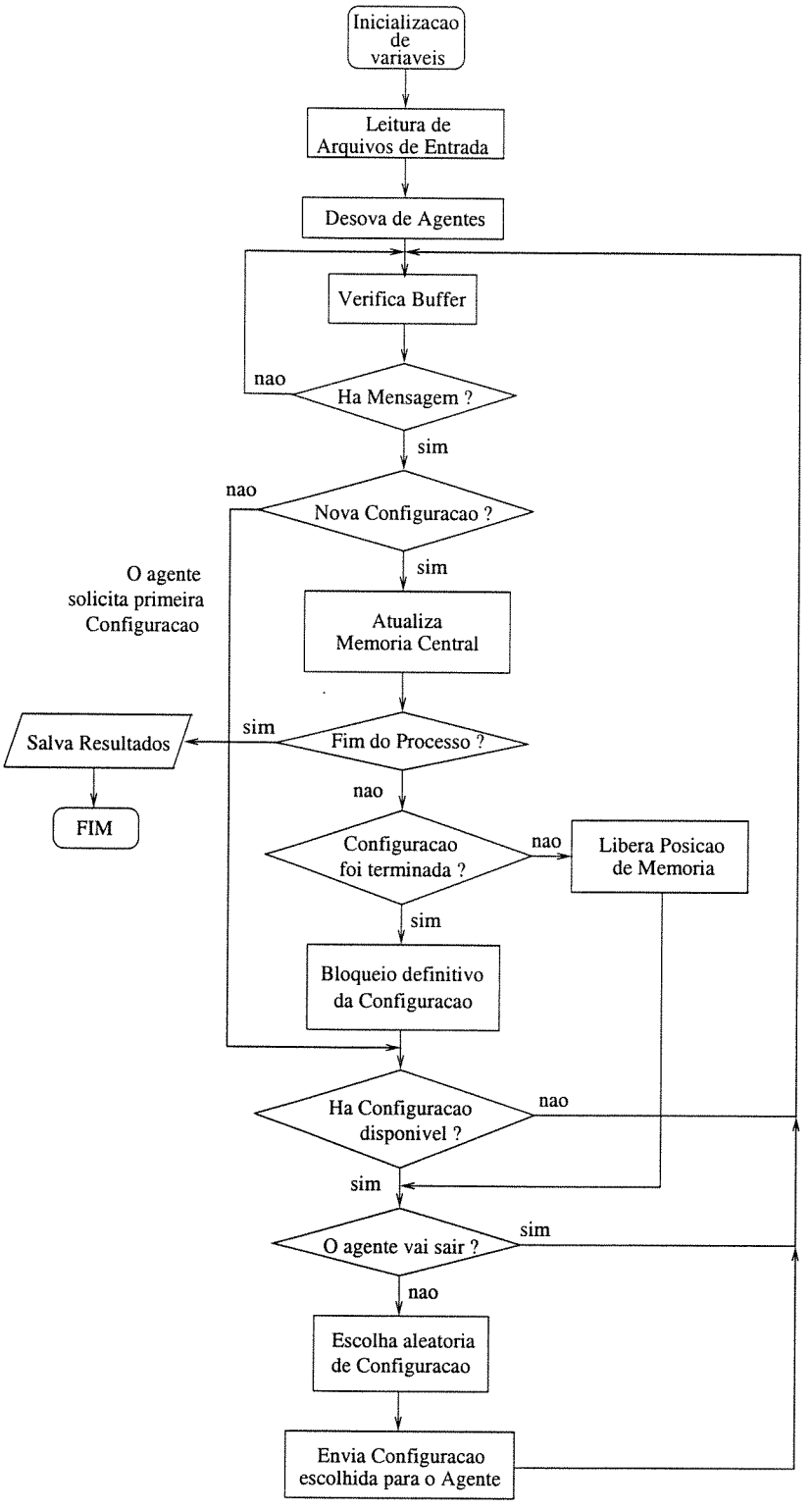


Figura 4.2: Estrutura do Programa *inic.*

4.4.3 *GARVERA*

O programa *GarverA* pode ser descrito dividindo-se o mesmo em etapas, assim como em *inic*:

1. Inicialização de variáveis e comandos iniciais do *PVM*.

   Nesta etapa inicial, o agente é incluído no grupo dinâmico [GEIS94] denominado "iniciar". Relembrando, todos os processos do Time-A obrigatoriamente farão parte deste grupo, para que possam ser identificados pelo seu *rank*, valor de identificação utilizado na remoção de agentes. Dentro desta mesma etapa, o agente também obtém o *ID* do processo *inic* dentro da MVP, uma vez que o *rank* do mesmo vale zero sempre.

2. Chama rotina que executa a leitura dos dados e parâmetros do problema.

   Dentro desta rotina, são lidos os valores contidos nos arquivos de dados relacionados anteriormente: *rede.dat*, *param.dat*, *nlin.dat* e *nconec.dat*.

3. Chama rotina de análise de rede pela primeira vez.

   Após a leitura dos dados do sistema e do problema do planejamento, é feita a análise da rede, isto é, o cálculo do fluxo de potência do sistema, através de uma rotina implementada por Gallego [GALL97], onde é feito um modelamento DC com programação linear. Para a resolução do problema utilizou-se o *MINOS*, um software destinado a cálculos de otimização. Neste ponto do programa, a análise é feita sem considerar qualquer alteração sobre a rede original. As alterações que os agentes efetuarem serão consideradas nas análises de rede futuras, dentro do *loop* principal, como será visto mais adiante.

4. Executa cálculos e ordenamentos inerentes ao método de Garver.

5. Envia sinal para *inic* solicitando primeira configuração.

   Esta é uma etapa importante do programa, pois é quando o agente efetivamente começa o seu trabalho, retirando um elemento da população para aplicar o seu método, produzindo uma nova configuração. Isto é feito enviando-se um sinal para o processo *inic*, avisando-o de que não está enviando uma nova configuração, pois acabou de entrar para o time. Posteriormente isto não será mais necessário pois cada vez que o agente colocar um novo elemento na Memória Central, ao mesmo tempo já retira outro para trabalhar automaticamente, desde que haja algum disponível.

6. Início do *loop* principal do programa.

   Aqui, o agente aguarda até que haja mensagem no *buffer* proveniente de *inic* ou de *remv*.

7. Verificação de mensagem recebida.

   Considerando que a mensagem enviada seja proveniente de *inic*, pode-se ter duas possibilidades. Numa primeira situação, tem-se uma configuração enviada para atualização (o que funciona como se o agente tivesse retirado um elemento da população). Num segundo caso, tem-se um sinal de aviso de encerramento da Memória Central, o que significa que não

há mais configurações disponíveis. Daí o agente deixa o grupo dinâmico e, em seguida, a MVP. A outra possibilidade é que a mensagem tenha vindo de *remv*, avisando que o agente deve se retirar do time.

8. Cálculos e ordenamentos inerentes ao método de Garver.

   Neste trecho entram os cálculos de análise de rede já considerando os efeitos das alterações efetuadas pelos agentes, isto é, o estado atual da configuração retirada da Memória Central. Aqui serão feitos também os cálculos e ordenamentos intrínsecos ao método de Garver.

9. Verificação do critério de parada.

   Cada metodologia utilizada em cada agente tem seu próprio critério de parada, isto é, *quando* a configuração atingiu seu estado final. Aqui, no método de Garver, compara-se o ramo que possui maior sobrecarga com um valor previamente estipulado em *param.dat*. Se o critério for satisfeito, então o programa vai para 14. Caso contrário, é feita a inclusão de linha ou caminho na configuração.

10. Inclusão de linha ou caminho na configuração.

    Aqui, inicialmente, é feita uma identificação de caminhos, isto é, se há a necessidade da inclusão de um caminho completo ao invés de apenas uma linha no ramo escolhido pela ordenação. Após esta identificação tem-se então a inclusão do caminho ou apenas uma linha na configuração retirada na Memória Central.

11. Envio de configuração atualizada para *inic*.

    Esta passagem corresponde a ação do agente de colocar um novo elemento de volta na Memória Central. Caso o agente tenha, numa última mensagem recebida, recebido um sinal de *remv* para encerramento, o mesmo enviará um sinal para *inic* junto com esta nova configuração comunicando a sua saída do time, evitando que *inic* mande outra configuração de volta (isto é, o agente pára de buscar elementos na Memória Central).

12. Verifica sinal de encerramento.

    Caso *remv* tenha mandado anteriormente um sinal para saída do agente, o programa sai do grupo dinâmico e em seguida deixa a MVP.

13. Retorna para a etapa 6.

14. Efetua refinamento da configuração atualizada.

    Aqui é feita uma análise da configuração terminada, a fim de se eliminar possíveis linhas ou caminhos em excesso, ou seja, linhas adicionadas desnecessariamente pelos agentes durante o processo do Inicializador.

15. Cálculo do custo da configuração obtida.

16. Envio de configuração terminada para *inic*.

    Esta configuração em seu estado final será rearmazenada em *mempop* para não mais ser alterada. Juntamente com a configuração, é enviado um sinal para *inic* avisando-o de que esta deve ser definitivamente bloqueada.

17. Verifica sinal de encerramento.

    Caso *remv* tenha mandado anteriormente um sinal para saída do agente, o programa sai do grupo dinâmico e em seguida deixa a MVP.

18. Retorna para a etapa 6.

Na Fig. 4.3 é apresentado um fluxograma onde pode-se visualizar melhor o funcionamento do processo *garverA*, segundo a descrição acima. Modelo este que serve também para os outros agentes.

4.4.4 *REMV*

O programa *remv* é um programa simples, cuja função é retirar um determinado agente do Time-A. Na verdade, este programa apenas manda um sinal para o agente em questão, ordenando a sua saída. Inicialmente, *remv* se adiciona ao grupo dinâmico "iniciar", para poder se fazer uso do sistema de *rank*. A seguir, é lido do console da *workstation* o valor de *rank* correspondente ao agente a ser retirado do time. A partir do valor lido, descobre-se o correspondente *ID* do processo desejado dentro da MVP. Assim, basta mandar uma mensagem contendo um sinal característico que será interpretado pelo agente como uma ordem de se retirar do Inicializador. Finalmente, *remv* sai do grupo dinâmico e da MVP, liberando o *prompt* da estação.

4.4.5 Observações Importantes

Após a descrição em detalhes de cada componente do Inicializador, algumas observações sobre o comportamento do Time-A são necessárias :

Loop Dedicado

Cada *loop* em *inic* é dedicado exclusivamente a um único agente, correspondendo ao processo no qual o agente "deposita" um novo elemento (configuração) na memória compartilhada (Memória Central) e em seguida retira um outro para novo processamento. Observa-se que, desde o instante em que *inic* verifica seu *buffer* até o momento no qual envia nova configuração, estes dois passos mais os outros intermediários são relativos a um mesmo agente, que necessariamente fica ocioso durante esse intervalo de tempo. Ao mesmo tempo também ficarão ociosos todos os agentes que mandaram mensagens durante este período, pois estas ficarão no *buffer* de *inic* e só serão verificadas, em ordem de chegada, após o término do *loop* relativo ao primeiro agente. Essa característica é inerente ao próprio modo como o time assíncrono foi implementado, e gera um sincronismo obrigatório, que controla o acesso à memória compartilhada, evitando que esta seja acessada por dois processos simultaneamente.

Número Máximo de Agentes

Uma outra característica, também ligada à programação do time, diz respeito ao número máximo possível de agentes. Dois agentes não podem trabalhar sobre os mesmos dados simultaneamente, e *inic* foi programado para retirar um dado agente quando não houver um elemento disponível para o mesmo, o que é lógico pois, o que o agente vai ficar fazendo se não há o que fazer? Disso conclui-se que, se em um dado instante, o número de agentes for superior ao número de configurações não definitivamente bloqueadas, estes agentes em excesso serão gradativamente eliminados por *inic*, de tal forma que o time sempre se ajusta para que haja um agente para cada elemento da população. Esta situação pode acontecer em três casos particulares :

1. No início do Time-A, se o número inicial de agentes for superior ao número de configurações em *param.dat.*
2. Durante o processamento do time, à medida que configurações forem se tornando definitivamente bloqueadas, o número de agentes é automaticamente reduzido para o número de configurações ainda em processo.
3. Durante o processamento do time, se novos agentes forem manualmente incluidos, de forma que o total de agentes seja maior que o número de configurações não definitivamente bloqueadas.

Obs.: É claro que podem haver menos agentes que configurações. Apenas o contrário é impossível.

Modelo Mestre/Escravo

Mesmo considerando-se *Times Assíncronos* como um modelo de programação paralela, cabe aqui ressaltar que em princípio, dentre os modelos tradicionais de processamento paralelo, utilizou-se o modelo *Mestre/Escravo.* Isto fica claro ao lembrar-se que inicialmente roda-se *inic* e este por sua vez executa processos diferentes em estações diferentes, onde todos rodam em paralelo. Claro que não é uma situação única, pois o time pode ser executado sem agentes iniciais e, neste caso, cada agente deverá ser adicionado manualmente.

O Tamanho do Grão

Em princípio, o que se espera de um Time-A é que não haja qualquer sincronismo entre os agentes, ou seja, que todos trabalhem sem qualquer dependência uns dos outros, sem que nenhum atrapalhe o trabalho de outro. Mas infelizmente não é o que acontece em particular no Inicializador proposto neste trabalho. Isto devido a granularidade destes processos agentes, que é muito fina, pois são poucos os cálculos. Vale lembrar que estão sendo utilizados Métodos Heurísticos Construtivos, ou, como costuma-se chamar, "Métodos Aproximados", que não são algoritmos de otimização "pesados" como Algoritmos Genéticos. Daí tem-se que os agentes estão quase que constantemente tentando acessar a Memória Central e, como não é possível que

dois agentes acessem ao mesmo tempo, acaba-se formando uma fila para que haja o acesso de maneira organizada. Com isso, surge um sincronismo entre os processos, pois o acesso de um à memória compartilhada, depende do término do acesso de outro, o que gera uma ociosidade dos agentes. Pode-se dizer que isto é um "defeito" do Inicializador, mas é algo inevitável dada a natureza dos algoritmos implementados. Contudo, vale ressaltar que esse sincronismo não elimina a característica básica de um time assíncrono de possuir *Agentes Autônomos*, veja logo a seguir nesta seção.

As Características de Time-A

Não há dúvidas que o Time-A proposto para o Inicializador neste trabalho realmente é um time assíncrono, pois possui as três características fundamentais, além de ser *scale efficient*. Os agentes apresentados são autônomos, pois cada um trabalha independentemente do outro, de forma que agentes podem ser removidos ou adicionados sem interferir no desempenho dos outros. As comunicações são assíncronas, pois os agentes acessam a Memória Central sem qualquer sincronismo entre si, não havendo nenhuma dependência lógica neste acesso, a não ser integridade dos dados. O fluxo de dados no time é cíclico, uma vez que os agentes lêem, modificam e armazenam informações continuamente na Memória Central, de forma que as soluções geradas por uns podem ser reutilizadas pelos outros. Por último, o Inicializador é *scale efficient*, pois ele é aberto (cresce facilmente), onde todos os agentes cooperam efetivamente de maneira que, quanto mais agentes, melhor o desempenho (respeitando-se o limite máximo de agentes, de acordo com o total de elementos da população).

Como Identificar os Agentes ?

Nas descrições dos membros do Inicializador vistas anteriormente, falou-se várias vezes em *rank* do processo, valor necessário para a remoção de agentes do Time-A. Mas como saber o valor de *rank* de um agente? Isto é feito através de arquivos auxiliares gerados por cada processo do time. O processo *inic* gera um arquivo de saída contendo o *ID* de cada agente desovado especificado em *opcoes*. Cada agente, por sua vez, gera um arquivo exclusivo contendo o seu *ID* e seu valor de *rank*. Com estes arquivos é possível identificar cada agente do Time-A para que este possa ser removido.

inic é um Gerenciador ou Administrador?

Não, pois suas atribuições são: funcionar como uma memória compartilhada e como um agente destruidor. *inic* não possui qualquer poder sobre *como* os agentes atuam ou operam dentro do time assíncrono.

O Agente Destruidor

Fato curioso é que o agente Destruidor, ao contrário dos outros agentes, não é representado por um processo UNIX isolado, mas sim por uma tarefa interna de *inic*, que também representa a Memória Central. A ação destruidora do agente está na sobreposição dos valores de *mempop*, na memória de *inic*, cada vez que um novo valor calculado é enviado por um agente. Isso garante

com que o número total de elementos da população permaneça constante.

Mais Sobre o Time Proposto

No Inicializador descrito neste capítulo parte-se do zero, ou seja, não são utilizadas configurações obtidas por outros meios como ponto de partida para o inicializador. Isso faz sentido uma vez que este Time-A por si só tem como tarefa gerar elementos iniciais para o problema do planejamento da expansão. Pode-se dizer que o Inicializador consiste em uma estratégia de inicialização para um outro Time-A (Algoritmos Genéticos, *Simulated Annealing*, Busca Tabu) que o utilizaria como gerador de soluções iniciais.
Quanto à política de seleção, os agentes escolhem elementos na Memória Central de maneira aleatória, como se viu na descrição do funcionamento da Memória Central, onde *inic* gera um número aleatório que corresponde ao elemento de *mempop* que será enviado ao agente que está a espera.
A política de destruição escolhida também é simples e está "amarrada" ao modo de operação da memória compartilhada, no que diz respeito ao acesso exclusivo das posições da mesma. Dois agentes não podem retirar o mesmo elemento da população. Consequentemente, não há sentido em armazenar-se um novo elemento retirado inicialmente de uma dada posição (que passa a estar bloqueada por esse agente), em outra posição diferente da inicial, o que provocaria perda de informação. Mais ainda, como as soluções geradas partem do zero rumo a uma determinada configuração, considera-se que cada novo valor obtido por um dos agentes constitui uma solução melhor que a anterior. Isto porque possui um número de linhas a serem adicionadas mais próximo do desejado. Tendo-se isto em mente, considerou-se como critério de destruição de soluções a simples sobreposição dos valores antigos na Memória Central.

Capítulo 5

Testes e Resultados

5.1 Sistema Garver de 6 Barras com Redespacho

O Sistema Garver de 6 Barras é um sistema amplamente conhecido e utilizado por pesquisadores para testes em novos algoritmos, por se tratar de um sistema pequeno e cuja solução ótima é conhecida. Nos testes realizados neste trabalho, foi permitido o redespacho da geração, daí utilizou-se o valor de capacidade máxima de geração do sistema como valor de geração no arquivo de dados. Os dados deste sistema de pequeno porte são apresentados no Apêndice B. Testes foram realizados para diversas estruturas do Time-A proposto, conforme pode-se observar nas tabelas e gráficos a seguir, variando-se os agentes utilizados, o número de agentes e o número de estações de trabalho utilizadas. Em cada simulação utilizou-se apenas um agente para cada estação de trabalho. Vale ressaltar também que, em cada execução do Inicializador, a sua estrutura foi mantida constante durante todo o processamento, isto é, agentes não foram nem removidos nem adicionados, apesar do programa permitir tal possibilidade. Para este sistema, foram feitas dez tomadas de tempo para cada estrutura diferente do Time-A, e a partir daí calculadas as médias correspondentes, utilizadas na análise comparativa de desempenho entre as estruturas. Dentre as diversas soluções iniciais encontradas pelo Inicializador, obteve-se a configuração ótima para este sistema, valor que já era previamente conhecido, embora não seja este o objetivo deste Time-A, que é a obtenção de configurações iniciais diversificadas e em tempo reduzido para a resolução do planejamento da expansão via métodos combinatoriais. A configuração ótima do Sistema Garver é dada por :

Linhas adicionadas:

$n_{03-05} = 1$, $n_{04-06} = 3$.
Custo das Linhas $= 110$
Corte de carga $= 0{,}0$

Tabela 5.1: Times-A com apenas um agente.

Simulação	Tempos (s)					
	1-A	1-B	1-C	1-D	1-E	1-F
1	13,621	13,125	32,133	9,737	10,868	7,816
2	13,081	13,462	26,509	10,373	11,119	7,017
3	13,058	12,614	25,615	12,857	8,868	7,008
4	12,454	12,661	29,395	9,861	13,553	7,711
5	12,421	12,624	29,702	9,323	11,414	8,399
6	12,708	12,632	30,084	10,557	11,177	7,916
7	12,579	12,551	27,078	9,284	12,172	7,584
8	12,608	12,568	28,375	10,821	11,683	6,549
9	12,778	12,650	28,840	11,089	9,858	7,201
10	12,455	12,642	32,517	10,219	11,238	7,580
Médias	12,776	12,753	29,025	10,412	11,195	7,478

Tabela 5.2: Times-A com dois agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	2-A	2-C	2-D	2-F
1	7,097	15,862	6,925	4,291
2	6,842	15,364	5,955	4,552
3	7,032	13,627	5,237	4,178
4	6,968	15,777	5,689	3,825
5	6,706	17,053	5,879	3,716
6	6,812	15,275	5,890	4,052
7	6,834	13,809	6,164	3,937
8	6,793	15,346	5,371	5,600
9	6,831	15,278	6,611	3,994
10	6,841	15,337	6,589	4,711
Médias	6,876	15,273	6,031	4,286

Tabela 5.3: Times-A com três agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	3-A	3-C	3-D	3-F
1	6,100	11,072	4,848	3,395
2	5,313	10,811	4,521	3,206
3	5,325	10,172	4,336	3,135
4	5,403	10,399	4,426	2,816
5	5,560	10,702	3,866	3,110
6	4,976	10,844	3,848	3,275
7	5,157	10,509	4,185	3,220
8	4,999	13,908	4,884	3,162
9	5,210	12,369	4,879	3,336
10	5,344	10,768	5,711	3,114
Médias	5,339	11,155	4,550	3,177

Tabela 5.4: Times-A com quatro agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	4-A	4-C	4-D	4-F
1	4,702	10,00	3,773	3,293
2	4,341	9,306	3,834	3,108
3	4,464	7,449	3,334	2,477
4	4,202	8,523	3,761	2,495
5	4,504	9,382	3,397	2,885
6	4,607	9,079	5,933	3,041
7	4,178	9,088	3,746	2,703
8	4,514	7,591	4,033	2,814
9	4,192	9,292	4,040	2,732
10	4,281	8,599	3,669	2,604
Médias	4,399	8,831	3,952	2,815

Tabela 5.5: Times-A com cinco agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	5-A	5-C	5-D	5-F
1	4,084	9,819	4,382	2,940
2	3,972	7,441	4,510	3,110
3	4,536	7,541	3,952	2,492
4	3,845	7,756	3,071	2,475
5	4,265	9,087	3,160	2,717
6	4,239	7,966	3,804	2,559
7	3,949	8,158	3,966	3,014
8	4,350	7,658	3,940	2,635
9	4,067	8,655	4,762	2,707
10	4,938	7,594	3,981	2,948
Médias	4,225	8,168	3,953	2,760

Tabela 5.6: Times-A com dois agentes.

Simulação	Tempos (s)			
	A/B	A/C	B/C	A/E
1	7,422	9,827	9,582	6,722
2	6,818	10,614	9,375	7,017
3	6,955	8,964	10,886	6,301
4	7,078	10,651	12,272	6,862
5	7,008	10,368	9,199	6,369
6	6,955	10,268	9,390	5,291
7	6,945	12,481	9,562	6,395
8	7,248	10,402	10,734	5,727
9	6,974	10,609	10,837	6,141
10	6,829	9,356	12,136	7,158
Médias	7,023	10,354	10,397	6,398

Tabela 5.7: Times-A com dois agentes.

Simulação	Tempos (s)				
	B/D	C/D	A/F	C/F	E/F
1	7,333	9,450	6,722	7,850	4,619
2	5,921	8,918	4,912	5,814	3,760
3	5,838	9,002	5,110	7,612	5,016
4	6,842	7,539	5,172	7,211	4,298
5	5,518	8,995	5,653	6,428	5,239
6	6,164	8,980	5,185	7,379	4,939
7	6,429	9,157	5,462	7,245	4,330
8	5,578	7,925	4,070	6,203	4,925
9	6,395	9,379	6,013	9,038	4,439
10	5,980	8,870	5,315	5,701	4,512
Médias	6,200	8,822	5,361	7,048	4,608

Tabela 5.8: Times-A com três agentes.

Simulação	Tempos (s)					
	A/B/C	A/C/D	A/C/F	A/B/E	A/B/F	C/D/F
1	9,659	6,409	6,884	7,719	5,406	5,854
2	9,841	7,399	7,496	5,404	4,734	5,476
3	7,428	8,494	5,964	5,523	5,137	5,378
4	9,740	6,383	7,278	5,735	5,945	5,920
5	9,953	7,398	5,713	5,477	5,690	5,637
6	9,608	7,365	6,482	5,538	5,968	6,343
7	12,805	6,728	7,356	5,596	5,393	5,750
8	6,635	7,020	7,326	5,169	5,654	5,872
9	7,567	5,884	7,252	6,326	4,855	7,408
10	6,774	7,581	7,604	5,131	5,552	5,597
Médias	9,001	7,066	6,936	5,762	5,433	5,924

Tabela 5.9: Times-A com quatro agentes.

Simulação	Tempos (s)		
	A/B/C/D	A/B/C/F	A/B/D/F
1	7,004	6,829	3,711
2	7,305	6,852	5,623
3	7,062	4,768	3,483
4	6,810	4,592	3,820
5	6,349	4,109	4,700
6	7,095	7,020	3,607
7	7,117	6,919	4,746
8	6,833	6,721	4,247
9	7,060	7,674	4,501
10	6,656	5,195	3,742
Médias	6,929	6,068	4,218

Tabela 5.10: Time-A com cinco agentes.

Simulação	Tempos (s)
	A/B/C/D/F
1	4,163
2	4,411
3	4,901
4	3,108
5	3,527
6	4,402
7	6,025
8	4,488
9	4,388
10	4,491
Médias	4,390

Tabela 5.11: Time-A com seis agentes.

Simulação	Tempos (s) A/B/C/D/E/F
1	4,369
2	4,523
3	6,130
4	4,481
5	3,058
6	4,392
7	4,538
8	4,406
9	4,316
10	4,260
Médias	4,447

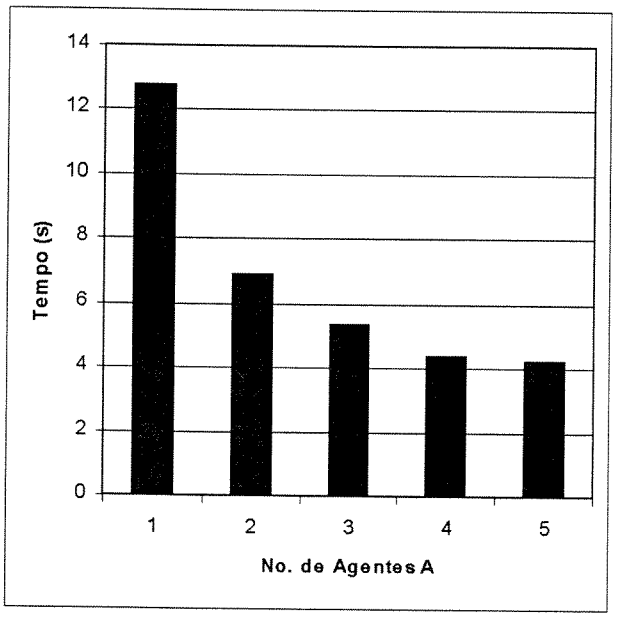


Figura 5.1: Times-A formados apenas pelo agente GarverA.

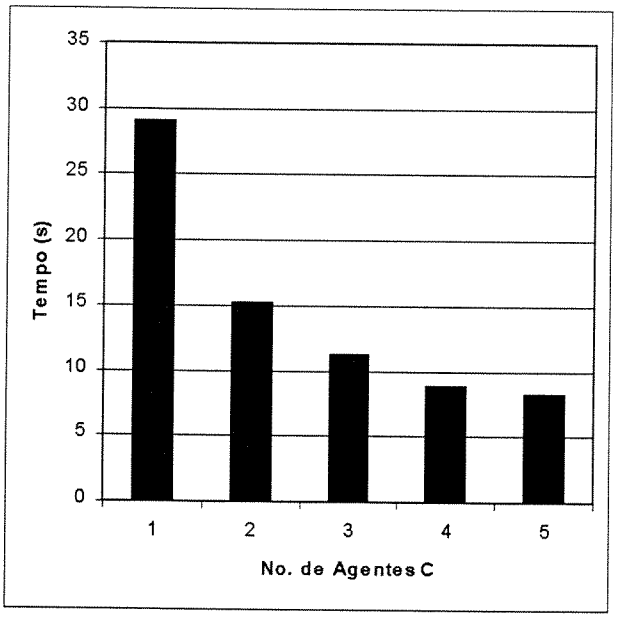


Figura 5.2: Times-A formados apenas pelo agente GarverC.

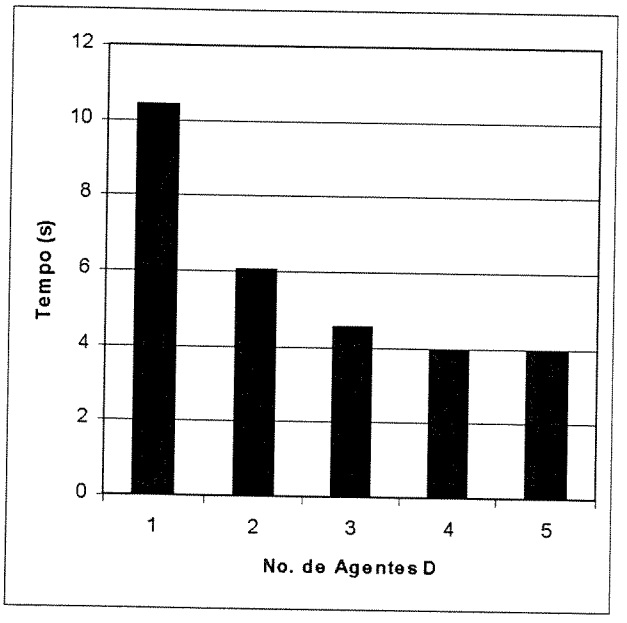


Figura 5.3: Times-A formados apenas pelo agente minD.

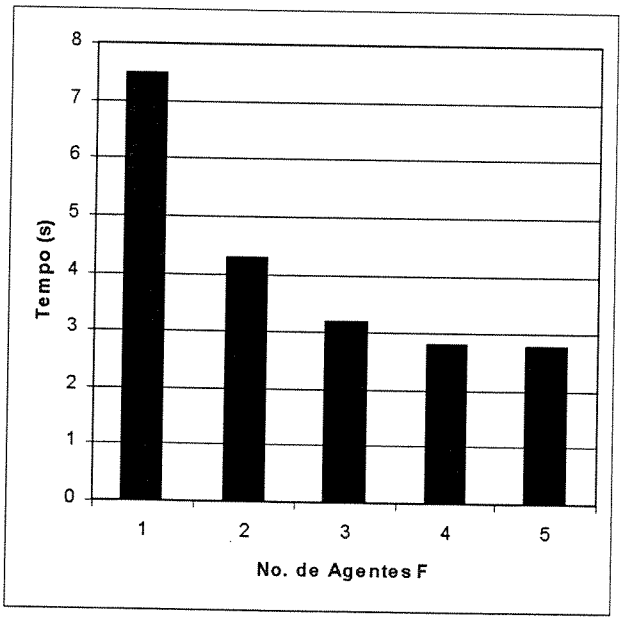


Figura 5.4: Times-A formados apenas pelo agente mccF.

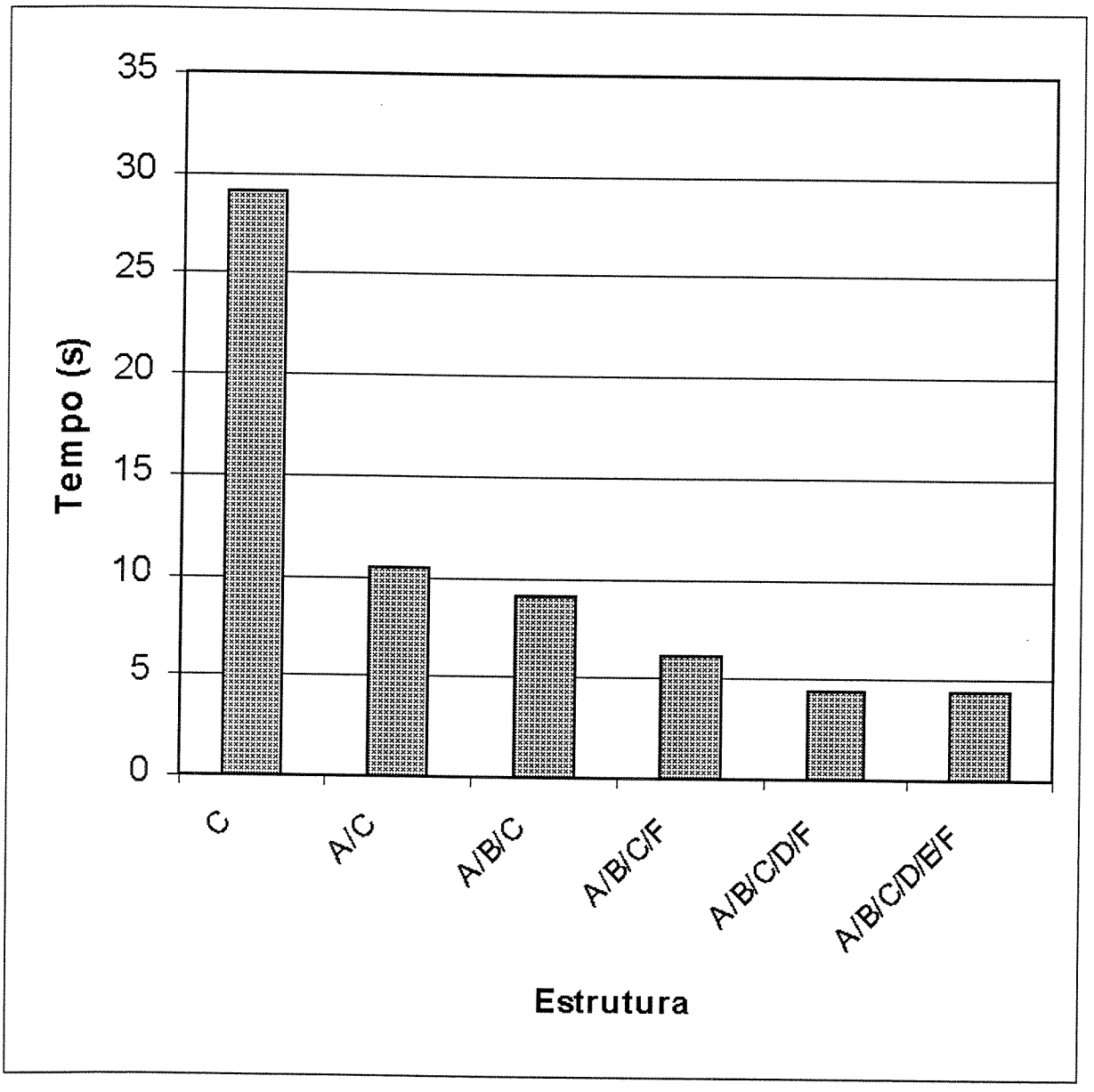


Figura 5.5: Algumas estruturas do Inicializador para o Sistema Garver.

Analisando as tabelas e os gráficos, observa-se que o Inicializador realmente apresenta uma característica *scale efficient*, pois os agentes realmente cooperam entre si eficientemente, reduzindo o tempo total de processamento, se bem que esta característica se mostrou de forma muito discreta, pois o sistema testado é muito pequeno, necessitando portanto, de pouco tempo computacional. Por isso, não foi possível obter um ganho muito grande em termos de redução de tempo, havendo muita comunicação entre os agentes e a Memória Central (assunto discutido no Cap.4) e consequentemente, uma saturação rápida da MVP. Isto pode ser observado nas Tabelas 5.9, 5.10 e 5.11, onde, apesar do aumento do número de agentes componentes do Time-A, não houve uma sensível redução do tempo total médio.

Observa-se também que o tempo de processamento não só varia com o número de agentes presentes no Time-A mas também dependendo de *quais* agentes formam o time. Por exemplo, pela Tabela 5.3, pode-se observar que os times formados por três agentes A ou três agentes D tiveram um tempo de processamento bem menor que o time formado por três agentes C, pois o agente GarverC é mais lento pelo próprio algoritmo. Na Tabela 5.8, nota-se que o time A/B/C demandou mais tempo de computação que o time A/C/D. Apesar de possuirem o mesmo número de agentes, estas duas estruturas do Inicializador também diferem no tempo gasto, pois seus agentes são diferentes.

5.2 Sistema Sul Brasileiro de 46 Barras com Redespacho

Aqui, testou-se o Sistema Sul Brasileiro para 1990 tomando 1980 como ano de referência. Este é um sistema de médio porte cuja configuração ótima também é conhecida. Para estes testes realizados, também foi permitido o redespacho da geração, mais uma vez utilizando-se o valor de capacidade máxima de geração do sistema como valor de geração no arquivo de dados para o Inicializador. Os dados deste sistema são apresentados no Apêndice B.

Testes foram realizados para diversas estruturas do Inicializador, conforme pode-se observar nas tabelas e gráficos seguintes, variando-se os agentes utilizados, o número de agentes e o número de estações de trabalho utilizadas, da mesma maneira que para o sistema anterior. As estruturas também foram mantidas constantes durante cada processamento do Inicializador. Para este sistema, foram feitas cinco tomadas de tempo para cada estrutura do Time-A, e a partir daí calculadas as médias correspondentes, utilizadas na análise comparativa de desempenho das estruturas.

A várias configurações iniciais geradas pelo Inicializador nestes testes apresentaram-se muito boas como blocos construtivos atraentes para os algoritmos combinatoriais, obtendo-se soluções iniciais próximas da ótima do sistema. A configuração ótima da rede Sul é dada por :

Linhas adicionadas:

$n_{13-20} = 1$, $n_{20-23} = 1$, $n_{20-21} = 2$, $n_{42-43} = 1$, $n_{46-06} = 1$,
$n_{05-06} = 2$.
Custo das Linhas $= 70289$
corte de carga $= 1$ MW

Sendo este um sistema de maior porte, e que consequentemente demanda mais tempo de computação, foi possível observar com mais clareza a propriedade *scale efficient* a partir dos resultados apresentados nas tabelas e gráficos a seguir. Ao contrário do sistema Garver, estes resultados não apresentaram saturação, havendo um ganho em redução do tempo de processamento à medida que as configurações do Time-A aumentaram em número de agentes.

Tabela 5.12: Times-A com apenas um agente.

Simulação	Tempos (s)					
	1-A	1-B	1-C	1-D	1-E	1-F
1	533	499	2672	633	781	919
2	536	492	2601	620	809	933
3	533	496	2589	572	780	900
4	534	494	2778	554	756	906
5	534	497	2511	574	774	930
Médias	534	495,6	2630,2	590,6	780	917,6

Tabela 5.13: Times-A com dois agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	2-A	2-C	2-D	2-F
1	270	1381	313	515
2	279	1348	290	483
3	271	1262	323	482
4	275	1423	244	489
5	276	1342	305	483
Médias	274,2	1351,2	295	490,4

Tabela 5.14: Times-A com três agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	3-A	3-C	3-D	3-F
1	215	938	206	349
2	231	853	248	302
3	220	912	206	340
4	215	915	222	335
5	216	1009	163	346
Médias	219,4	925,4	209	334,4

Tabela 5.15: Times-A com quatro agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	4-A	4-C	4-D	4-F
1	142	761	157	289
2	154	734	173	279
3	149	742	186	255
4	147	770	190	275
5	148	732	165	272
Médias	148	747,8	174,2	274

Tabela 5.16: Times-A com cinco agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	5-A	5-C	5-D	5-F
1	127	527	126	192
2	126	615	136	194
3	115	569	138	175
4	115	521	137	183
5	116	560	126	198
Médias	119,8	558,4	132,6	188,4

Tabela 5.17: Times-A com seis agentes iguais em estações diferentes.

Simulação	Tempos (s)			
	6-A	6-C	6-D	6-F
1	96	428	110	187
2	95	476	110	169
3	99	428	146	171
4	96	470	105	173
5	96	479	116	185
Médias	96,4	456,2	117,4	177

Tabela 5.18: Times-A com dois agentes.

Simulação	Tempos (s)			
	A/B	A/C	B/C	A/E
1	260	522	469	398
2	257	510	469	450
3	285	523	467	332
4	270	509	467	418
5	267	523	466	402
Médias	267,8	517,4	467,6	400

Tabela 5.19: Times-A com dois agentes.

Simulação	Tempos (s)				
	B/D	C/D	A/F	C/F	E/F
1	354	433	509	771	401
2	333	480	501	663	354
3	322	466	502	705	367
4	301	474	507	670	398
5	307	377	500	720	403
Médias	323,4	446	503,8	705,8	384,6

Tabela 5.20: Times-A com três agentes.

Simulação	Tempos (s)					
	A/B/C	A/C/D	A/C/F	A/B/E	A/B/F	C/D/F
1	253	291	488	252	343	302
2	251	283	449	252	323	325
3	242	257	481	243	320	244
4	252	285	496	273	339	284
5	251	271	479	271	308	301
Médias	249,8	277,4	478,6	258,2	326,6	291,2

Tabela 5.21: Times-A com quatro agentes.

Simulação	Tempos (s)		
	A/B/C/D	A/B/C/F	A/B/D/F
1	246	305	202
2	192	304	205
3	218	298	166
4	235	330	218
5	214	300	199
Médias	221	307,4	198

Tabela 5.22: Time-A com cinco agentes.

Simulação	Tempos (s)
	A/B/C/D/F
1	196
2	196
3	210
4	202
5	190
Médias	198,8

Tabela 5.23: Time-A com seis agentes.

Simulação	Tempos (s)
	A/B/C/D/E/F
1	150
2	153
3	187
4	173
5	164
Médias	165,4

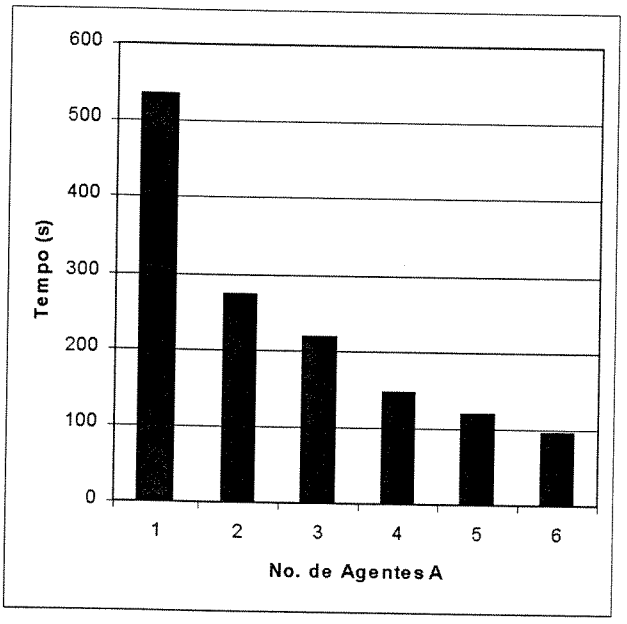


Figura 5.6: Times-A formados apenas pelo agente GarverA.

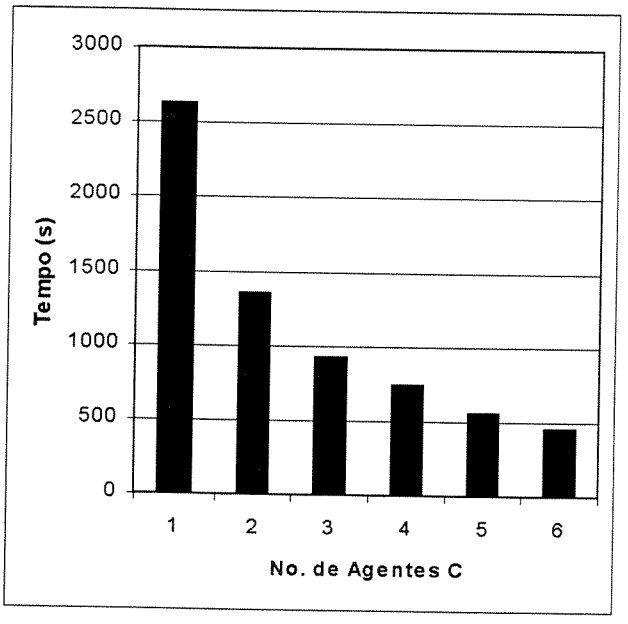


Figura 5.7: Times-A formados apenas pelo agente GarverC.

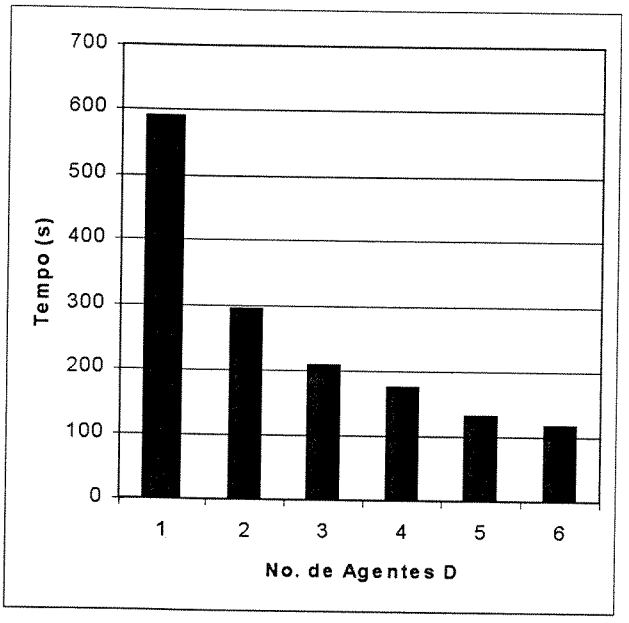


Figura 5.8: Times-A formados apenas pelo agente minD.

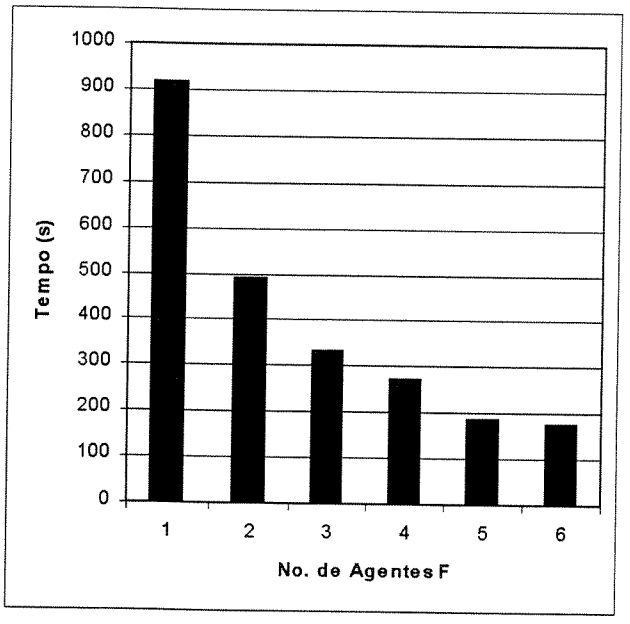


Figura 5.9: Times-A formados apenas pelo agente mccF.

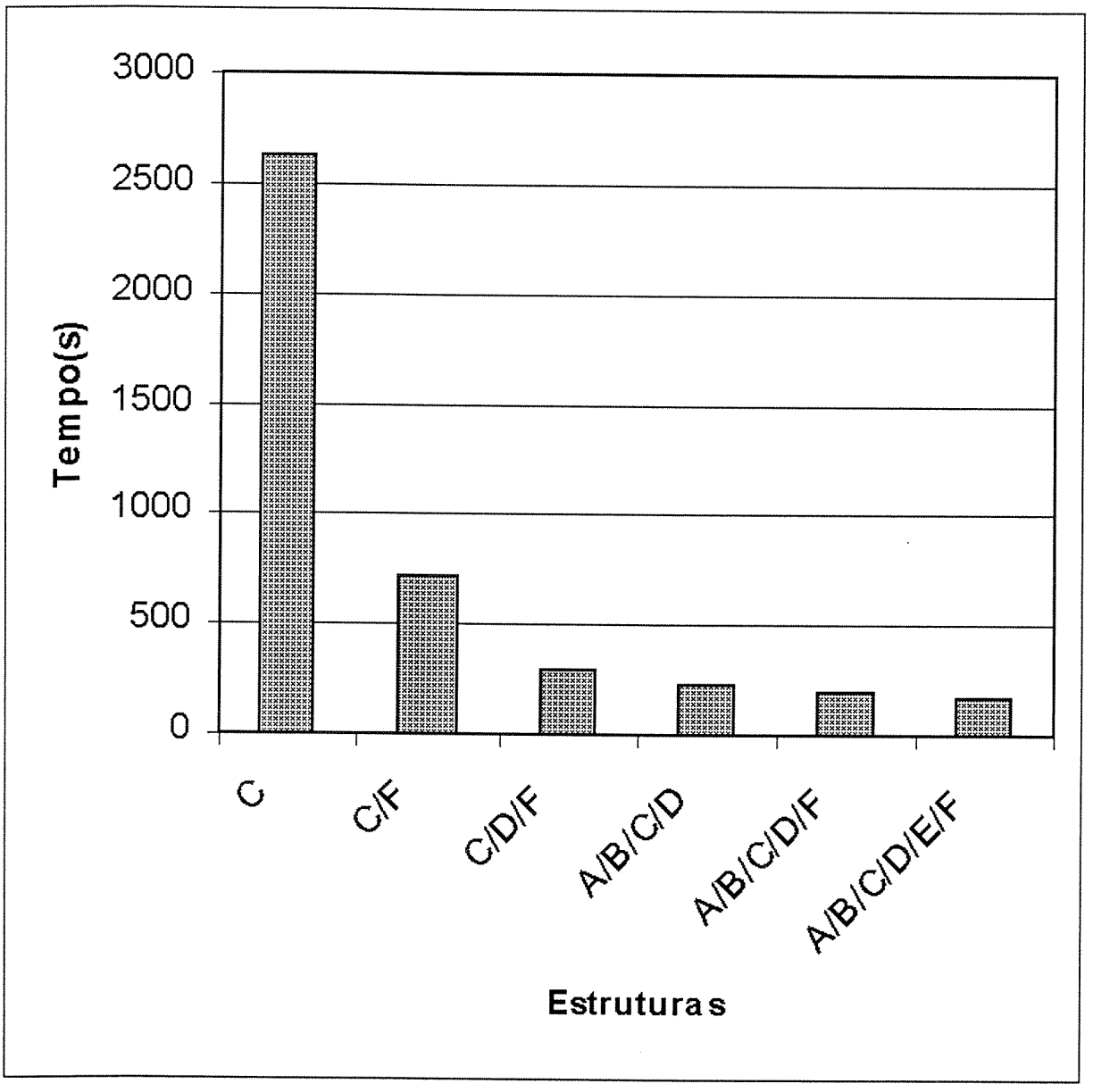


Figura 5.10: Algumas estruturas do Inicializador para o Sistema Sul.

5.3 Sistema Norte-Nordeste Brasileiro de 87 Barras

O Sistema Norte-Nordeste é um sistema de difícil resolução, por apresentar um grande número de barras isoladas em sua configuração inicial, ou seja, possui elevado nível de ilhamento. Para os testes realizados neste trabalho, considerou-se uma previsão para o ano 2008 com referência em 1980. Os dados da configuração inicial deste sistema são apresentados no Apêndice B. Para este sistema de grande porte, tem-se da literatura que o mesmo não foi resolvido usando técnicas de otimização exatas, mas sim apenas métodos aproximados. Por isso, ainda não se conhece sua solução ótima. Para as análises deste trabalho, será considerada a melhor configuração obtida por Gallego [GALL97], que é dada por :

Linhas adicionadas:

$n_{01-02} = 1$, $n_{02-04} = 1$, $n_{04-05} = 3$, $n_{04-06} = 2$, $n_{04-81} = 3$,
$n_{15-16} = 1$, $n_{05-56} = 1$, $n_{05-58} = 3$, $n_{06-70} = 1$, $n_{12-13} = 1$,
$n_{14-59} = 5$, $n_{15-46} = 4$, $n_{16-44} = 2$, $n_{16-61} = 6$, $n_{18-50} = 1$,
$n_{13-15} = 11$, $n_{18-74} = 5$, $n_{00-66} = 2$, $n_{22-37} = 2$, $n_{22-58} = 1$,
$n_{24-25} = 1$, $n_{25-26} = 3$, $n_{26-27} = 1$, $n_{26-27} = 1$, $n_{27-53} = 1$,
$n_{28-35} = 2$, $n_{34-39} = 1$, $n_{35-46} = 1$, $n_{35-47} = 1$, $n_{35-51} = 3$,
$n_{36-39} = 1$, $n_{36-46} = 1$, $n_{39-42} = 2$, $n_{39-86} = 1$, $n_{40-46} = 2$,
$n_{41-64} = 2$, $n_{47-48} = 5$, $n_{48-50} = 2$, $n_{52-59} = 1$, $n_{53-76} = 1$,
$n_{53-86} = 1$, $n_{54-58} = 1$, $n_{56-57} = 1$, $n_{58-78} = 2$, $n_{62-72} = 1$,
$n_{63-64} = 1$, $n_{65-66} = 3$, $n_{68-83} = 1$, $n_{70-82} = 1$, $n_{71-75} = 2$,
$n_{71-83} = 1$, $n_{73-75} = 1$.
Custo das Linhas = 2625663
corte de carga = 0,0

Também aqui testou-se o sistema para várias estruturas do Time-A, sempre mantendo-se as mesmas constantes durante o processamento. Porém neste caso, foi efetuada apenas uma tomada de tempo para cada estrutura testada. Isso se deve ao fato da rede Nordeste ser de difícil solução, levando a grandes tempos de computação, e isto faz com que os tempos obtidos para cada estrutura sejam nitidamente diferentes. Graças a isto, apesar do valor de tempo variar a cada execução, não foi necessária uma média de valores para a análise comparativa.
E através destes valores obtidos, mais uma vez constatou-se a característica *scale efficient* do Inicializador. Para esta rede, obteve-se configurações iniciais razoavelmente atraentes. Não se poderia esperar resultados melhores, uma vez que o sistema é muito complexo e ao mesmo tempo os métodos conseguem apenas resultados aproximados. A Tab. 5.25 apresenta valores da porcentagem de acertos do Inicializador em relação a melhor configuração obtida em [GALL97], para algumas estruturas testadas. Para se compreender melhor, por exemplo, uma configuração que possua 50% de acerto significa que metade dos seus elementos coincidem com os elementos obtidos na melhor solução.

Tabela 5.24: Tempos obtidos para o Sistema Nordeste.

Estrutura	Tempo (s)
1-A	23660
2-A	11920
3-A	8229
4-A	6251
5-A	5730
6-A	4374
A/B	12651
A/B/C	10267
A/B/C/D	6483
A/B/C/D/F	4987
A/B/C/D/E/F	3230

Tabela 5.25: Porcentagem de acertos em relação à melhor configuração obtida

Estrutura	%	Estrutura	%
A	73,46	C/D	72,91
B	74,30	C/E	65,92
C	68,16	C/F	52,23
D	74,30	D/E	74,30
E	68,44	A/B/C	69,27
F	56,98	A/B/E	59,22
A/B	79,89	A/B/C/D/E	72,07
A/C	74,86	A/A/B/B/C/C	70,39
A/D	71,23	A/A/B/B/C/D/E	68,44
B/C	72,91		

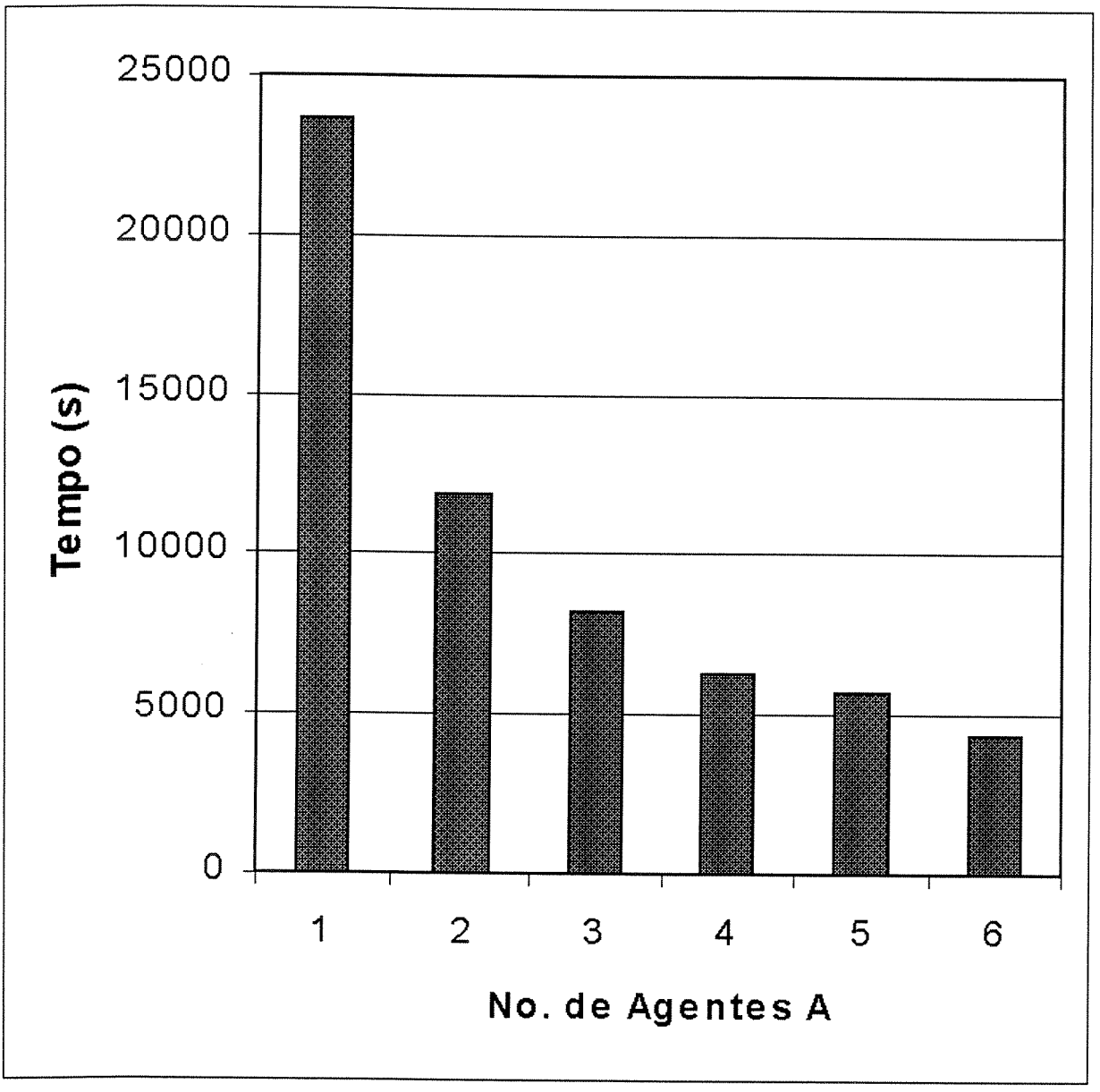


Figura 5.11: Times-A formados apenas pelo agente GarverA.

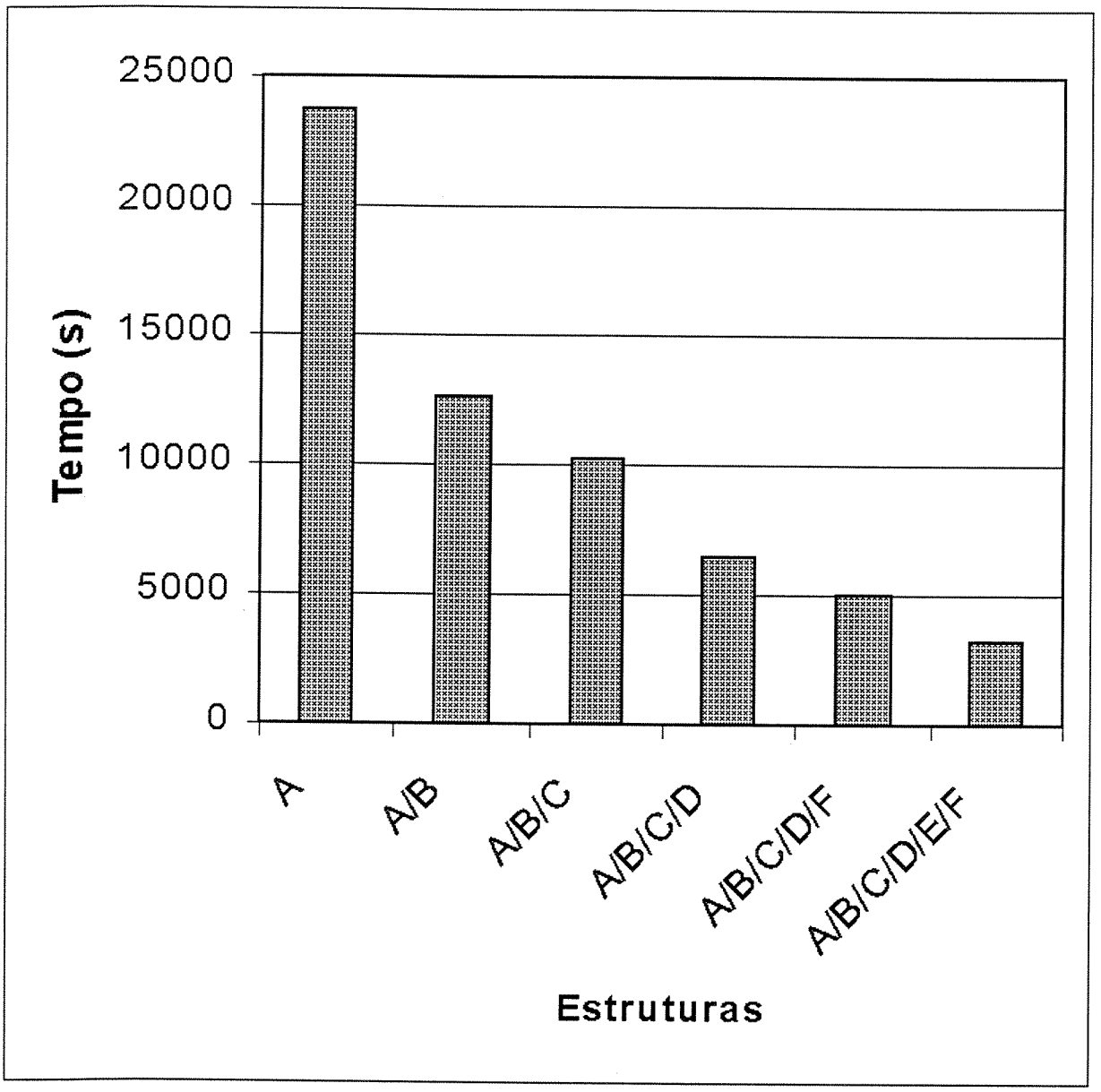


Figura 5.12: Algumas estruturas do Inicializador para o Sistema Nordeste.

Capítulo 6

Conclusões

Para o estudo do problema do planejamento da expansão da transmissão, normalmente modela-se a rede pelo fluxo de carga DC, o que leva o problema de planejamento estático a um PNLIM. Dadas as dimensões que o problema assume para casos práticos em geral observa-se o fenômeno da explosão combinatorial (problema NP-completo). Métodos heurísticos como Algoritmos Genéticos e Busca Tabu exigem uma inicialização adequada, normalmente através da determinação prévia de soluções iniciais atraentes. Com a tendência cada vez maior do uso de redes de computadores nas concessionárias de energia elétrica, aliado a disponibilidade de bibliotecas para processamento paralelo, tem-se o ambiente propício para a utilização de modelos de programação distribuída.
Particularmente, neste trabalho, foram aplicados os conceitos que envolvem o modelo de Times-A aos algoritmos heurísticos construtivos (Garver, Mínimo Esforço e Mínimo Corte de Carga), para a implementação de *programas inicializadores.*

A partir dos testes e resultados apresentados, conclui-se que o trabalho teve seus objetivos plenamente alcançados, que foram os seguintes :

- Estudar, compreender e implementar os fundamentos teóricos da metodologia de Times Assíncronos;
- Obter soluções iniciais factíveis para o problema do planejamento da expansão da transmissão em menor tempo;
- Diversificar as soluções iniciais obtidas.

Neste trabalho, procurou-se obter soluções iniciais para o planejamento de tal forma que estas se aproximassem, se possível, das soluções ótimas ou da melhor solução obtida por métodos combinatoriais. Porém, como foi visto anteriormente, o Inicializador tem como objetivo a obtenção de soluções iniciais factíveis diversificadas em menor tempo de processamento. Mais

ainda, soluções obtidas nos testes que foram consideradas como satisfatórias ou boas, podem não sê-lo de fato. Isto porque funções objetivo podem ter vários ótimos locais mas apenas um ótimo global. Métodos como o de Garver são eficientes para a obtenção de soluções próximas de um ótimo local (que eventualmente pode ser o próprio global). Porém, uma solução próxima de um ótimo local pode estar muito distante do ótimo global. Aí entra o fator diversificação do Inicializador, pois, uma configuração encontrada por este que aparentemente é ruim, pode ser útil para métodos como os Algoritmos Genéticos, que precisam de soluções iniciais diversificadas para aumentar as chances de se encontrar a solução global do problema.

O modelo de Times-A realmente mostrou-se perfeitamente programável em locais onde se trabalha com redes de *workstations* ou mesmo PCs e com linguagens de fácil uso como o FORTRAN, desde que se tenha em mãos ferramentas para a paralelização dos problemas, como o *PVM*. Sendo assim, obteve-se um conjunto de programas que, trabalhando em conjunto, formam um time onde seus membros trabalham de maneira autônoma, sem que um interfira no trabalho do outro. Membros podem ser removidos ou adicionados a qualquer momento, durante o processamento do time. Todos cooperam mutuamente, sempre trabalhando em paralelo e assincronamente, de maneira a reduzir o tempo total de processamento. Cada componente desta equipe utiliza valores calculados pelos outros componentes, o que constitui um fluxo de dados cíclico.

Sugestões para futuros trabalhos

- Aperfeiçoamento dos algoritmos utilizados;

  Aqui propõe-se variações dos métodos construtivos utilizados. Acredita-se que, com variações dos critérios de parada, do cálculo dos índices de sensibilidade, ou dos métodos de ordenamento dos ramos, possa-se alcançar soluções mais atrativas.

- Outras estruturas de Times-A;

  A estrutura de Times-A utilizada neste trabalho é das mais simples. Outras possíveis estruturas, combinando-se os agentes de maneira que estes comuniquem entre si, ou mesmo criando-se outras memórias compartilhadas, podem melhorar o desempenho do Inicializador.

- Testes em outros ambientes (diferentes máquinas, diferentes bibliotecas paralelas).

  Neste trabalho testes foram feitos em uma rede homogênea de estações de trabalho SUN/SPAR Propõe-se a realização de testes em uma rede heterogênea, onde haja a presença de estações mais rápidas ou mesmo de arquiteturas diferentes, como a IBM. A paralelização dos programas foi feita pelo uso das bibliotecas do software PVM. Outra versão do Inicializador poderia ser implementada utilizando-se as bibliotecas MPI que, assim como o PVM, é de domínio público.

Referências Bibliográficas

[BAYO94] Bayona G., Pérez Arriaga J.I. and CHOPIN, "A Heuristic Model for Long Term Transmission Expansion Planning", Presented at the IEEE/PES Winter Meeting, New York, NY, Jan./Feb. 1994.

[DSOU91] de Souza P.S. and Talukdar S.N., "Genetic Algorithms in Asynchronous Teams", Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos:CA, 1991.

[DSOU93] de Souza P.S., "Asynchronous Organizations for Multialgorithm Problems", Ph.D. dissertation, Dep. of Electrical and Computer Engineering. Carnegie Mellon University. April 1993.

[DUSO71] Dusonchet Y.P., Talukdar S.N. and Sinnot H.E., "Load Flow using a combination of Point Jacobi and Newton's Methods", IEEE Transactions on Power Apparatus and Systems, PAS-90, p.941-949, 1971.

[FOX 79] Fox M.S., "Organizational Structuring: Designing Large, Complex Software", Technical Report CMU-CS-79-155, Carnegie Mellon University, 1979.

[GALL97] Gallego R.A., "Planejamento a Longo Prazo de Sistemas de Transmissão Usando Técnicas de Otimização Combinatorial", Tese de Doutorado em Engenharia Elétrica, Faculdade de Engenharia Elétrica e de Computação , UNICAMP, 1997.

[GARV70] Garver L.L., "Transmission Network Estimation Using Linear Programming", IEEE Transactions on Power Apparatus and Systems, Vol.PAS-89, p.1688-1697, Sept./Oct. 1970.

[GEIS94] Geist A. *et al.*, "PVM 3 User's Guide and Reference Manual", Oak Ridge: Oak Ridge National Laboratory, May 1994.

[GRAN85] Granville S. and Pereira M.V.F, "Analysis of the Linearized Power Flow Model in Benders Decomposition", EPRI-Report RP 2473-6, Stanford University, Feb. 1985.

[MEHR83] Mehrotra R. and Talukdar S.N., "Task Scheduling on Multiprocessors for Power System Problems", IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, No. 11, Nov. 1983.

[MONT82] Monticelli A. *et al*, "Interactive Transmission Network Planning Using a Least-Effort Criterion", IEEE Transactions on Power Apparatus and Systems, Vol.PAS-101, No.10, p.3919-3925, Oct. 1982.

[MURT87] Murtagh B.A., Saunders M.A., "MINOS 5.1 User's Guide", Stanford: Department of Operations Research of Stanford University, 1987. (Technical Report SOL 83-20R).

[OLIV95] Oliveira G.C., Costa A.P. and Binato S., "Large Scale Transmission Network Planning Using Optimization and Heuristic Techniques", Presented at the IEEE/PES Winter Meeting, New York, NY, Jan./Feb. 1995.

[ORTE68] Ortega J.M. and Rheinboldt W.C., "Local and Global convergence of generalized linear iterations", Symposium on Numerical Solution of Nonlinear Problems. SIAM, Filadelfia, Pennsylvania,1968.

[OSTE78] Oster G.F. and Wilson E.O., "Caste and Ecology in the Social Insects", Princeton University Press, Princeton, NJ, 1978.

[PERE85a] Pereira M.V.F., Pinto L.M.V.G., "Application of Sensitivity Analysis of Load Supplying Capability to Interactive Transmission Expansion Planning", IEEE Transactions on Power Apparatus and Systems, Vol.PAS-104, No.2, p.381-389, Feb. 1985.

[PERE85b] Pereira M.V.F., Pinto L.M.V.G., Cunha S.H.F. and Oliveira G.C., "A Decomposition Approach to Automated Generation/Transmission Expansion Planning", IEEE Transactions on Power Apparatus and Systems, Vol.PAS-104, No.11, Nov. 1985.

[PERE87] Pereira M.V.F., Pinto L.M.V.G., Oliveira G.C. and Cunha S.H.F., "Composite Generation-Transmission Expansion Planning", EPRI-Research Project 2473-9, Stanford University, 1987.

[PYO 85] Pyo S.S., "Asynchronous Algorithms for Distributed Processing", Ph.D. dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, 1985.

[RAME91] Ramesh V.C. *et al.*, "Asynchronous Teams. An Organizational Model for Distributed Problem Solving", 1991 Summer National Meeting, American Institute of Chemical Engineers, Pittsburgh.

[RAME96] Ramesh V.C. and Talukdar S.N., "A Parallel Asynchronous Decomposition for On-Line Contingency Planning", IEEE Transactions on Power Systems, Vol.11, No. 1, Feb. 1996.

[RODR94] Rodrigues M., Saavedra O.R. and Monticelli A., "Asynchronous Programming Model for The Concurrent Solution of The Security Constrained Optimal Power Flow Problem", IEEE Transactions on Power Systems, Vol.9, No.4, Nov. 1994.

[ROME94] Romero R. and Monticelli A., "A Hierarchical Decomposition Approach for Transmission Network Expansion Planning", IEEE Transactions on Power Systems, Vol.9, No.1, Feb. 1994.

[SHAF87] Shafritz J.M., Ott J.S., "Classics of Organization Theory", Dorsey Press, Chicago, IL, 1987.

[TALU83a] Talukdar S.N., Pyo S.S. and Giras T.C., "Asynchronous Procedures for Parallel Processing", IEEE Transactions on Power Apparatus and Systems,Vol.PAS-102, No.11, Nov. 1983.

[TALU83b] Talukdar S.N., Pyo S.S. and Mehrotra R., "Designing Algorithms and Assignments for Distributed Processing", Electric Power Research Institute, Palo Alto, CA, 1983.

[TALU85] Talukdar S.N. and Cardozo E., "Artificial Intelligence Techniques for Power System Operations. Some Demonstration and an Assessment of Opportunities.", EPRI Report No. 1999-7. 1985.

[TALU90] Talukdar S.N. and de Souza P.S., "Asynchronous Teams", Second SIAM Conf. on Linear Algebra: Signals, Systems, and Control, San Francisco, CA, Nov. 1990.

[TALU91] Talukdar S.N., Ramesh V.C. and Nixon J.C., "A Distributed System of Control Specialist for Real-Time Operations", Proceeding of the 3^{rd} Symposium on Expert Systems Applications to Power Systems. Japan. 1991.

[TALU92a] Talukdar S., Ramesh V.C., Quadrel R. and Christie R., "Multiagent Organizations for Real-Time Operations", Proceedings of the IEEE, vol.80, No.5, May 1992.

[TALU92b] Talukdar S.N. and de Souza P.S., "Scale Efficient Organizations", IEEE Int. Conference on Systems, Man and Cybernetics, Chicago, IL, Oct. 1992.

[TALU93a] Talukdar S.N. and de Souza P.S., "Objects Organizations and Super-Objects", Engineering Design:the creation of products and processes, McGraw Hill, 1993.

[TALU93b] Talukdar S.N. and Ramesh V.C., "A parallel global optimization algorithm and its application to the CCOPF problem", Proceedings of the Power Industry Computer Applications, Phoenix, May 1993.

[TALU94] Talukdar S. and Ramesh V.C., "A Multi-Agent Technique for Contingency Constrained Optimal Power Flows", IEEE Transactions on Power Systems, Vol. 9, No. 2, May 1994.

[VILL85] Villanasa R., Garver L.L. and Salon S.J., "Transmission Network Planning Using Linear Programming", IEEE Transactions on Power Apparatus and Systems,Vol.PAS-104, No.2, Feb. 1985.

Apêndice A

Primitivas do PVM Utilizadas

PVMFBUFINFO()

Sintaxe

```
call PVMFBUFINFO(bufid, bytes, msgtag, tid, info)
```

Discussão

A rotina *PVMFBUFINFO* retorna informações sobre o *buffer* referido em *bufid*. No trabalho, esta rotina foi utilizada apenas com a finalidade de se obter o valor de *msgtag*, que é uma etiqueta de identificação, que na verdade representa o tipo do conteúdo da mensagem recebida, de forma que não seja necessária a leitura da mesma através de *PVMFUNPACK*.

PVMFEXIT()

Sintaxe

```
call PVMFEXIT(info)
```

Discussão

A rotina *PVMFEXIT* avisa ao *pvmd* local que este processo está deixando o *PVM*. Esta rotina está presente em todos os programas que compõem o Inicializador.

PVMFGETTID()

Sintaxe

```
call PVMFGETTID(group, inum, tid)
```

Discussão

A rotina *PVMFGETTID* retorna o ID do processo *PVM* pertencente ao grupo dinâmico *group* e cujo *instance number* vale *inum*. Esta rotina está presente nos agentes, pois estes precisam descobrir o ID de *inic*; e em *remv*, pois este precisa encontrar o ID do processo a ser removido do Time-A.

PVMFINITSEND()

Sintaxe

```
call PVMFINITSEND(encoding, bufid)
```

Discussão

A rotina *PVMFINITSEND* apaga o *buffer* identificado por *bufid* e o prepara para o empacotamento de uma nova mensagem. Esta rotina está presente em todos os programas que compõem o Inicializador, uma vez que todos enviam dados.

PVMFJOINGROUP()

Sintaxe

```
call PVMFJOINGROUP(group, inum)
```

Discussão

A rotina *PVMFJOINGROUP* adiciona o processo que a chamou ao grupo dinâmico *group* e retorna sua identificação dentro do grupo em *inum*. Esta rotina está presente em todos os programas que compõem o Inicializador.

PVMFLVGROUP()

Sintaxe

```
call PVMFLVGROUP(group, info)
```

Discussão

A rotina *PVMFLVGROUP* faz com que o processo que a chamou seja retirado do grupo dinâmico identificado por *group*. Esta rotina está presente em todos os programas que compõem o Inicializador, pois todos pertencem a um mesmo grupo dinâmico.

PVMFMYTID()

Sintaxe

```
call PVMFMYTID(tid)
```

Discussão

A rotina *PVMFMYTID* é a primeira rotina *PVM* que deve ser chamada em um programa que pretende pertencer a uma MVP. Ela acrescenta o processo à máquina virtual paralela retornando um único ID para o mesmo, através do qual o mesmo processo poderá se comunicar com os outros processos da mesma MVP.

PVMFNRECV()

Sintaxe

```
call PVMFNRECV(tid, msgtag, bufid)
```

Discussão

A rotina *PVMFNRECV* verifica se chegou mensagem com rótulo (etiqueta) *msgtag* proveniente de *tid*. Em caso afirmativo, *PVMFNRECV* imediatamente coloca a mensagem em um novo *buffer* de recepção ativo, apagando o atual, retornando seu identificador em *bufid*. É uma rotina do tipo não bloqueante, ou seja, após a verificação, o fluxo do programa segue independentemente da chegada ou não de nova mensagem. *PVMFNRECV* está presente apenas em *inic*.

PVMFPACK()

Sintaxe

```
call PVMFPACK(what, xp, nitem, stride, info)
```

Discussão

A rotina *PVMFPACK* empacota variáveis, vetores ou matrizes de um determinado tipo para serem enviadas a outro processo. Esta rotina está presente em todos os programas que compõem

o Inicializador.

PVMFPARENT()

Sintaxe

```
call PVMFPARENT(tid)
```

Discussão

A rotina *PVMFPARENT* retorna o ID do processo "pai" deste processo que a chamou. Esta rotina está presente em todos os programas que compõem o Inicializador.

PVMFRECV ()

Sintaxe

```
call PVMFRECV(tid, msgtag, bufid)
```

Discussão

A rotina *PVMFRECV* possui a mesma função que *PVMFNRECV*, com a diferença de se tratar de uma rotina bloqueante, ou seja, o processo é bloqueado até que a mensagem com as especificações determinadas chegue no *buffer*. Esta rotina está presente em todos os programas que compõem o Inicializador.

PVMFSEND()

Sintaxe

```
call PVMFSEND(tid, msgtag, info)
```

Discussão

A rotina *PVMFSEND* envia uma mensagem armazenada no *buffer* de envio ativo para o processo *PVM* identificado por *tid*. Esta rotina está presente em todos os programas que compõem o Inicializador.

PVMFSPAWN()

Sintaxe

```
call PVMFSPAWN(task, flag, where, ntask, tids, numt)
```

Discussão

A rotina *PVMFSPAWN* executa *ntask* cópias do executável *task* nas estações determinadas por *flag* e *where*. Ao mesmo tempo, *PVMFSPAWN* retorna em *tids* os IDs dos processos desovados. Esta é a rotina responsável pela execução dos agentes iniciais determinados em *opcoes*.

PVMFUNPACK()

Sintaxe

```
call PVMFUNPACK(what, xp, nitem, stride, info)
```

Discussão

A rotina *PVMFUNPACK* desempacota o *buffer* de recepção ativo em matrizes, variáveis ou vetores de um determinado tipo. Esta rotina está presente em todos os programas que compõem o Inicializador.

Apêndice B

Dados dos Sistemas de Teste

B.1 Sistema Garver de 6 Barras

B.2 Sistema Sul Brasileiro de 46 Barras

B.3 Sistema Norte-Nordeste Brasileiro de 87 Barras

B.1 Sistema Garver de 6 Barras

Níveis de Geração e Carga

Barra	Cap. de Geração	Ger. Atual	Carga (MW)
1	150	50	80
2	0	0	240
3	360	165	40
4	0	0	160
5	0	0	240
6	600	545	0

Características das Linhas

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
1	1-2	1	0.40	100	40
2	1-3	0	0.38	100	38
3	1-4	1	0.60	80	60
4	1-5	1	0.20	100	20
5	1-6	0	0.68	70	68
6	2-3	1	0.20	100	20
7	2-4	1	0.40	100	40
8	2-5	0	0.31	100	31
9	2-6	0	0.30	100	30
10	3-4	0	0.59	82	59
11	3-5	1	0.20	100	20
12	3-6	0	0.48	100	48
13	4-5	0	0.63	75	63
14	4-6	0	0.30	100	30
15	5-6	0	0.61	78	61

B.2 Sistema Sul Brasileiro de 46 Barras

Níveis de Geração e Carga

Barra	Cap. de Geração	Ger. Atual	Carga (MW)
1	0.0	0.0	0.0
2	0.0	0.0	443.1
3	0.0	0.0	0.0
4	0.0	0.0	300.7
5	0.0	0.0	238.0
6	0.0	0.0	0.0
7	0.0	0.0	0.0
8	0.0	0.0	72.2
9	0.0	0.0	0.0
10	0.0	0.0	0.0
11	0.0	0.0	0.0
12	0.0	0.0	511.9
13	0.0	0.0	185.8
14	1257.0	944.0	0.0
15	0.0	0.0	0.0
16	2000.0	1366.0	0.0
17	1050.0	1000.0	0.0
18	0.0	0.0	0.0
19	1670.0	773.0	0.0
20	0.0	0.0	091.2
21	0.0	0.0	0.0
22	0.0	0.0	81.9
23	0.0	0.0	458.1
24	0.0	0.0	478.2
25	0.0	0.0	0.0

Níveis de Geração e Carga (Sul)

Barra	Cap. de Geração	Ger. Atual	Carga (MW)
26	0.0	0.0	231.9
27	220.0	54.0	0.0
28	800.0	730.0	0.0
29	0.0	0.0	0.0
30	0.0	0.0	0.0
31	700.0	310.0	0.0
32	500.0	450.0	0.0
33	0.0	0.0	229.1
34	748.0	221.0	0.0
35	0.0	0.0	216.0
36	0.0	0.0	90.1
37	300.0	212.0	0.0
38	0.0	0.0	216.0
39	600.0	221.0	0.0
40	0.0	0.0	262.1
41	0.0	0.0	0.0
42	0.0	0.0	607.9
43	0.0	0.0	0.0
44	0.0	0.0	79.1
45	0.0	0.0	86.7
46	700.0	599.0	0.0

Características das Linhas (Sul)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
1	01-07	1	0.0616	270	4349
2	01-02	2	0.1065	270	7076
3	04-09	1	0.0924	270	6217
4	05-09	1	0.1173	270	7732
5	05-08	1	0.1132	270	7480
6	07-08	1	0.1023	270	6823
7	04-05	2	0.0566	270	4046
8	02-05	2	0.0324	270	2581
9	08-13	1	0.1348	240	8793
10	09-14	2	0.1756	220	11267

Características das Linhas (Sul)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
11	12-14	2	0.0740	270	5106
12	14-18	2	0.1514	240	9803
13	13-18	1	0.1805	220	11570
14	13-20	1	0.1073	270	7126
15	18-20	1	0.1997	200	12732
16	19-21	1	0.0278	1500	32632
17	16-17	1	0.0078	2000	10505
18	17-19	1	0.0061	2000	8715
19	14-26	1	0.1614	220	10409
20	14-22	1	0.0840	270	5712
21	22-26	1	0.0790	270	5409
22	20-23	2	0.0932	270	6268
23	23-24	2	0.0774	270	5308
24	26-27	2	0.0832	270	5662
25	24-34	1	0.1647	220	10611
26	24-33	1	0.1448	240	9399
27	33-34	1	0.1265	270	8288
28	27-36	1	0.0915	270	6167
29	27-38	2	0.2080	200	13237
30	36-37	1	0.1057	270	7025
31	34-35	2	0.0491	270	3591
32	35-38	1	0.1980	200	12631
33	37-39	1	0.0283	270	2329
34	37-40	1	0.1281	270	8389
35	37-42	1	0.2105	200	13388
36	39-42	3	0.2030	200	12934
37	40-42	1	0.0932	270	6268
38	38-42	3	0.0907	270	6116
39	32-43	1	0.0309	1400	35957
40	42-44	1	0.1206	270	7934
41	44-45	1	0.1864	200	11924
42	19-32	1	0.0195	1800	23423
43	46-19	1	0.0222	1800	26365
44	46-16	1	0.0203	1800	24319
45	18-19	1	0.0125	600	8178
46	20-21	1	0.0125	600	8178
47	42-43	1	0.0125	600	8178
48	02-04	0	0.0882	270	5965
49	14-15	0	0.0374	270	2884
50	46-10	0	0.0081	2000	10889

Características das Linhas (Sul)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
51	04-11	0	0.2246	240	14247
52	05-11	0	0.0915	270	6167
53	46-06	0	0.0128	2000	16005
54	46-03	0	0.0203	1800	24319
55	16-28	0	0.0222	1800	26365
56	16-32	0	0.0311	1400	36213
57	17-32	0	0.0232	1700	27516
58	19-25	0	0.0325	1400	37748
59	21-25	0	0.0174	2000	21121
60	25-32	0	0.0319	1400	37109
61	31-32	0	0.0046	2000	7052
62	28-31	0	0.0053	2000	7819
63	28-30	0	0.0058	2000	8331
64	27-29	0	0.0998	270	6672
65	26-29	0	0.0541	270	3894
66	28-41	0	0.0339	1300	39283
67	28-43	0	0.0406	1200	46701
68	31-41	0	0.0278	1500	32632
69	32-41	0	0.0309	1400	35957
70	41-43	0	0.0139	2000	17284
71	40-45	0	0.2205	180	13994
72	15-16	0	0.0125	600	8178
73	46-11	0	0.0125	600	8178
74	24-25	0	0.0125	600	8178
75	29-30	0	0.0125	600	8178
76	40-41	0	0.0125	600	8178
77	02-03	0	0.0125	600	8178
78	05-06	0	0.0125	600	8178
79	09-10	0	0.0125	600	8178

B.3 Sistema Nordeste Brasileiro de 87 Barras

Níveis de Geração e Carga

Barra	Geração em 2008 (MW)	Carga em 2008 (MW)
1	0	2747
2	4550	0
3	0	0
4	6422	0
5	0	0
6	0	0
7	0	31
8	82	0
9	465	0
10	538	0
11	2260	0
12	4312	0
13	5900	0
14	542	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	125
20	0	181
21	0	1044
22	0	446
23	0	84
24	0	230
25	0	2273

Níveis de Geração e Carga (Nordeste)

Barra	Geração em 2008 (MW)	Carga em 2008 (MW)
26	0	68
27	0	546
28	0	273
29	0	68
30	0	273
31	0	225
32	0	0
33	0	0
34	0	107
35	1531	0
36	0	325
37	114	0
38	0	0
39	0	269
40	0	1738
41	0	752
42	0	494
43	0	0
44	0	5819
45	0	0
46	0	297
47	0	0
48	0	432
49	0	1124
50	0	7628
51	0	420
52	0	1024
53	0	0
54	0	0
55	0	0
56	0	0
57	0	0
58	0	0
59	0	0
60	0	0

Níveis de Geração e Carga (Nordeste)

Barra	Geração em 2008 (MW)	Carga em 2008 (MW)
61	0	0
62	0	0
63	0	0
64	0	0
65	0	0
66	0	0
67	1242	0
68	888	0
69	902	0
70	0	0
71	0	0
72	0	0
73	0	0
74	0	0
75	0	0
76	0	0
77	0	0
78	0	0
79	0	0
80	0	0
81	0	0
82	0	0
83	0	0
84	0	0
85	0	705
86	0	0
87	0	0

Características das Linhas (Nordeste)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
1	01-02	2	0.0374	1000	44056
2	02-04	0	0.0406	1000	48880
3	02-60	0	0.0435	1000	52230
4	02-87	1	0.0259	1000	31192
5	03-71	0	0.0078	3200	92253
6	03-81	0	0.0049	3200	60153
7	03-83	0	0.0043	3200	53253
8	03-87	0	0.0058	1200	21232
9	04-05	1	0.0435	1000	52230
10	04-06	0	0.0487	1000	58260
11	04-32	0	0.0233	300	7510
12	04-60	0	0.0215	1000	26770
13	04-68	0	0.0070	1000	10020
14	04-69	0	0.0162	1000	20740
15	04-81	0	0.0058	1200	21232
16	04-87	1	0.0218	1000	26502
17	05-06	1	0.0241	1000	29852
18	05-38	2	0.0117	600	8926
19	05-56	0	0.0235	1000	29182
20	05-58	0	0.0220	1000	27440
21	05-60	0	0.0261	1000	32130
22	05-68	0	0.0406	1000	48880
23	05-70	0	0.0464	1000	55580
24	05-80	0	0.0058	1200	21232
25	06-07	1	0.0288	1000	35212
26	06-37	1	0.0233	300	7510
27	06-67	0	0.0464	1000	55580
28	06-68	0	0.0476	1000	56920
29	06-70	0	0.0371	1000	44860
30	06-75	0	0.0058	1200	21232
31	07-08	1	0.0234	1000	29048
32	07-53	0	0.0452	1000	54240
33	07-62	0	0.0255	1000	31460
34	08-09	1	0.0186	1000	23420
35	08-12	0	0.0394	1000	47540
36	08-17	0	0.0447	1000	53570

Características das Linhas (Nordeste)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
37	08-53	1	0.0365	1200	44190
38	08-62	0	0.0429	1000	51560
39	08-73	0	0.0058	1200	21232
40	09-10	1	0.0046	1000	7340
41	10-11	1	0.0133	1000	17390
42	11-12	1	0.0041	1200	6670
43	11-15	1	0.0297	1200	36284
44	11-17	1	0.0286	1200	35078
45	11-53	1	0.0254	1000	31326
46	12-13	1	0.0046	1200	7340
47	12-15	1	0.0256	1200	31594
48	12-17	1	0.0246	1200	30388
49	12-35	2	0.0117	600	8926
50	12-84	0	0.0058	1200	21232
51	13-14	0	0.0075	1200	10690
52	13-15	0	0.0215	1200	26770
53	13-17	0	0.0232	1200	28780
54	13-45	1	0.0290	1200	35480
55	13-59	1	0.0232	1200	28780
56	14-17	0	0.0232	1200	28780
57	14-45	0	0.0232	1200	28780
58	14-59	0	0.0157	1200	20070
59	15-16	2	0.0197	1200	24760
60	15-45	0	0.0103	1200	13906
61	15-46	1	0.0117	600	8926
62	15-53	0	0.0423	1000	50890
63	16-44	4	0.0117	600	8926
64	16-45	0	0.0220	1200	27440
65	16-61	0	0.0128	1000	16720
66	16-77	0	0.0058	1200	21232
67	17-18	2	0.0170	1200	21678
68	17-59	0	0.0170	1200	21678
69	18-50	4	0.0117	600	8926
70	18-59	1	0.0331	1200	40170
71	18-74	0	0.0058	1200	21232
72	19-20	1	0.0934	170	5885

Características das Linhas (Nordeste)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
73	19-22	1	0.1877	170	11165
74	20-21	1	0.0715	300	6960
75	20-38	2	0.1382	300	12840
76	20-56	0	0.0117	600	8926
77	20-66	0	0.2064	170	12210
78	21-57	0	0.0117	600	8926
79	22-23	1	0.1514	170	9130
80	22-37	2	0.2015	170	11935
81	22-58	0	0.0233	300	7510
82	23-24	1	0.1651	170	9900
83	24-25	1	0.2153	170	12705
84	24-43	0	0.0233	300	7510
85	25-26	4	0.1073	300	29636
86	25-55	0	0.0117	600	8926
87	26-27	4	0.1404	300	25500
88	26-29	1	0.1081	170	6710
89	26-54	0	0.0117	600	8926
90	27-28	3	0.0826	170	5335
91	27-35	2	0.1367	300	25000
92	27-53	1	0.0117	600	8926
93	28-35	3	0.1671	170	9900
94	29-30	1	0.0688	170	4510
95	30-31	1	0.0639	170	4235
96	30-63	0	0.0233	300	7510
97	31-34	1	0.1406	170	8525
98	32-33	0	0.1966	170	11660
99	33-67	0	0.0233	300	7510
100	34-39	3	0.1160	170	7150
101	34-41	2	0.0993	170	6215
102	35-46	4	0.2172	170	12705
103	35-47	2	0.1327	170	8085
104	35-51	3	0.1602	170	9625
105	36-39	2	0.1189	170	7315
106	36-46	2	0.0639	170	4235
107	39-42	1	0.0973	170	6105
108	39-86	0	0.0233	300	7510

Características das Linhas (Nordeste)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
109	40-45	1	0.0117	600	8926
110	40-46	3	0.0875	170	5500
111	41-64	0	0.0233	300	7510
112	42-44	2	0.0698	170	4565
113	42-85	2	0.0501	170	3465
114	43-55	0	0.0254	1000	31326
115	43-58	0	0.0313	1000	38160
116	44-46	3	0.1671	170	10010
117	47-48	2	0.1966	170	11660
118	48-49	1	0.0757	170	4895
119	48-50	2	0.0256	170	2090
120	48-51	2	0.2163	170	12760
121	49-50	1	0.0835	170	5335
122	51-52	2	0.0560	170	3795
123	52-59	1	0.0117	600	8926
124	53-54	0	0.0270	1000	32120
125	53-70	0	0.0371	1000	44860
126	53-76	0	0.0058	1200	21232
127	53-86	0	0.0389	1000	46870
128	54-55	0	0.0206	1000	25028
129	54-58	0	0.0510	1000	60940
130	54-63	0	0.0203	1000	25430
131	54-70	0	0.0360	1000	43520
132	54-79	0	0.0058	1200	21232
133	56-57	0	0.0122	1000	16050
134	58-78	0	0.0058	1200	21232
135	60-66	0	0.0233	300	7510
136	60-87	0	0.0377	1000	45530
137	61-64	0	0.0186	1000	23420
138	61-85	0	0.0233	300	7510
139	61-86	0	0.0139	1000	18060
140	62-67	0	0.0464	1000	55580
141	62-68	0	0.0557	1000	66300
142	62-72	0	0.0058	1200	21232
143	63-64	0	0.0290	1000	35480
144	65-66	0	0.3146	170	18260

Características das Linhas (Nordeste)

No.	Linha	Linhas existentes	Reatância (pu)	Capacidade (MW)	Custo 10^3 dol.
145	65-87	0	0.0233	300	7510
146	67-68	0	0.0290	1000	35480
147	67-69	0	0.0209	1000	26100
148	67-71	0	0.0058	1200	21232
149	68-69	0	0.0139	1000	18060
150	68-83	0	0.0058	1200	21232
151	68-87	0	0.0186	1000	23240
152	69-87	0	0.0139	1000	18060
153	70-82	0	0.0058	1200	21232
154	71-72	0	0.0108	3200	125253
155	71-75	0	0.0108	3200	125253
156	71-83	0	0.0067	3200	80253
157	72-73	0	0.0100	3200	116253
158	72-83	0	0.0130	3200	149253
159	73-74	0	0.0130	3200	149253
160	73-75	0	0.0130	3200	149253
161	73-84	0	0.0092	3200	107253
162	74-84	0	0.0108	3200	125253
163	75-76	0	0.0162	3200	185253
164	75-81	0	0.0113	3200	131253
165	75-82	0	0.0086	3200	101253
166	75-83	0	0.0111	3200	128253
167	76-77	0	0.0130	3200	149253
168	76-82	0	0.0086	3200	101253
169	76-84	0	0.0059	3200	70953
170	77-79	0	0.0151	3200	173253
171	77-84	0	0.0115	3200	132753
172	78-79	0	0.0119	3200	137253
173	78-80	0	0.0051	3200	62253
174	79-82	0	0.0084	3200	98253
175	80-81	0	0.0101	3200	117753
176	80-82	0	0.0108	3200	125253
177	80-83	0	0.0094	3200	110253
178	81-83	0	0.0016	3200	23253
179	82-84	0	0.0135	3200	155253