**UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"**
**FACULDADE DE ENGENHARIA - CAMPUS DE ILHA SOLTEIRA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

# Algoritmos Busca Tabu Paralelos Aplicados ao Planejamento da Expansão da Transmissão de Energia Elétrica

**Elisângela Menegasso Mansano**

**Orientador: Prof. Dr. Sérgio Azevedo de Oliveira**

Dissertação apresentada à Faculdade de Engenharia de Ilha Solteira - UNESP, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Ilha Solteira - SP, 20 de Fevereiro de 2008

unesp

**UNIVERSIDADE ESTADUAL PAULISTA**
**CAMPUS DE ILHA SOLTEIRA**
**FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA**

## CERTIFICADO DE APROVAÇÃO

**TÍTULO:** Algoritmos de Busca Tabu Paralelos Aplicados ao Planejamento da Expansão da Transmissão de Energia Elétrica

**AUTORA: ELISÂNGELA MENEGASSO MANSANO**
**ORIENTADOR: Prof. Dr. SERGIO AZEVEDO DE OLIVEIRA**

Aprovada como parte das exigências para obtenção do Título de MESTRE em ENGENHARIA ELÉTRICA pela Comissão Examinadora:

Prof. Dr. SERGIO AZEVEDO DE OLIVEIRA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. JOSE ROBERTO SANCHES MANTOVANI
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. FUJIO SATO
Departamento de Sistemas de Energia Elétrica / Universidade Estadual de Campinas

Data da realização: 20 de fevereiro de 2008.

Presidente da Comissão Examinadora
Prof. Dr. SERGIO AZEVEDO DE OLIVEIRA

# Agradecimentos

A Deus por me dar saúde e vitalidade para concluir mais esta etapa.

A minha família, a meu pai Jesus, minha mãe Aparecida e meu irmão Cleber, pelo apoio e ajuda durante todas as fases do meu mestrado.

Ao Professor Dr. Sérgio Azevedo de Oliveira, pela compreensão e por ter se mostrado um exemplo de orientador, teve grande colaboração na elaboração deste trabalho.

A meu querido amado Silvio, pela sua grande ajuda e companheirismo... Obrigada pelo apoio e pela paciência.

Ao Professor Dr. Rubén Augusto Romero Lázaro, pelas dicas no decorrer de minhas apresentações.

Aos professores Shinoda e Mantovani. E também às secretárias do DEE e PPGEE.

Aos técnicos do laboratório de informática, Deoclécio e Beto, que me deram muito apoio e atenção.

A amiga Silvia Taglialenha pelo carinho e troca de informações e ao meu amigo Fernando Sanchez, pela grande ajuda em Linux e ao meu querido amigo Hélio.

E também agradeço a todos do DEE e do PPGEE que de uma forma ou de outra contribuíram para a conclusão deste trabalho.

**A Deus, e à minha família.**

**"Tantas vezes pensamos ter chegado,**
**tantas vezes é preciso ir além..."**
**Fernando Pessoa**

MANSANO, Elisângela Menegasso. **Algoritmos Busca Tabu Paralelos Aplicados ao Planejamento da Expansão da Transmissão de Energia Elétrica**. 2008. 128 f. Dissertação (Mestrado em Engenharia Elétrica - Automação) - Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, 2008.

# Resumo

Neste trabalho, aborda-se o uso da metaheurística Busca Tabu (BT) na resolução do Problema de Planejamento da Expansão da Transmissão (PPET), analisado sob o ponto de vista estático, com o desenvolvimento de algoritmos paralelos em ambiente MPI ("*Message Passing Interface*").

A metaheurística Busca Tabu, é uma técnica baseada em parâmetros de controle, a estrutura de vizinhança e seu próprio algoritmo com poderosas estratégias de busca. Nesta técnica, dada uma configuração, deseja-se passar ao melhor vizinho através da entrada e saída de ramos, obtendo assim a configuração incumbente. Com essa configuração que é considerada como a melhor configuração encontrada até o momento, e mesmo sendo um bom valor o sistema continua a busca procurando mais configurações até encontrar uma que seja melhor que as já encontradas até o momento.

As versões paralelas dos algoritmos foram desenvolvidas a partir de um algoritmo BT serial avançado, sob o paradigma de programação SPMD (*"Single Program, Multiple Data"*), e as mesmas foram testadas para sistemas testes de pequeno porte (Garver - 6 barras/15 ramos), médio porte (Sul brasileiro - 46 barras/79 ramos) e grande porte (Norte-Nordeste brasileiro - 87 barras/179 ramos) e seus resultados comparados com o resultado do algoritmo BT serial. Esta comparação mostrou que os algoritmos propostos obtiveram um melhor desempenho, com alta eficiência.

**Palavras chave:** Metaheurística, Planejamento da Expansão de Redes, Busca Tabu, Programação Paralela.

# Abstract

This paper deals with the use of Tabu Search metaheurístic applied to solving the problem of transmission system expansion planning (TSEP), analyzed on the static point of view, with the development of parallel algorithms in the environment MPI (?Message Passing Interface ").

Tabu Search metaheurístic is a technique based on the control parameters, the structure of the neighborhood and its own algorithm with powerful search strategies. In this technique, given a configuration we want to progress to the best neighbor across the entrance and exit of branches, so getting the configuration incumbent. With this configuration which is regarded as the best configuration found so far, and this is a very good value, the system continuously seeking more settings to find a better than those found throughout the search.

The parallel versions of the algorithms were developed from an advanced TS series algorithm on the paradigm of programming SPMD (Single Program Multiple Data), and they were tested for test systems small scale (Garver - bars 6/15 branches), medium scale (South Brazilian - 46 bars/79 branches) and large scale (North-Northeast Brazilian - 87 bars/179 branches), and their results compared with the result of the series algorithm TS. This comparison showed that the proposed algorithms obtained best performance and high efficiency.

**Keywords:** Metaheurístic, Planning Expansion Network, Tabu Search, Parallel Programming.

# *Lista de Figuras*

# *Lista de Tabelas*

# *Lista de Abreviaturas, Variáveis e Símbolos*

A nomenclatura dos parâmetros usados em todos os sistemas testados foram:

| | |
|---|---|
| **(kseed)** | número aleatório; |
| **(malea)** | proposta aleatória; |
| **(mcarg)** | proposta de carga; |
| **(md1)** | constante para diversificar por adição e retirada; |
| **(md2)** | constante para diversificar por residência; |
| **(melhor)** | melhores configurações armazenadas; |
| **(mgarv)** | proposta de Garver; |
| **(mgera)** | proposta de geração; |
| **(mmin1)** | proposta de mínimo esforco sobrecarga; |
| **(mmin2)** | proposta de mínimo esforco índice; |
| **(msens)** | proposta de sensibilidade; |
| **(mtrono)** | número de trocas (que não melhoraram no passado); |
| **(mtrosi)** | número de trocas (que melhoraram no passado); |
| **(nactmax2)** | limitante do número de filhos; |
| **(nafa)** | custo de racionamento; |
| **(nanaliza)** | número máximo de ocasiões em que se analisa com diversificação; |
| **(ncl)** | número máximo de caminhos a serem gerados; |
| **(ncorte)** | mínimo valor do corte de carga permitido; |
| **(ncosto)** | proposta por custo para sair; |
| **(ndiver1)** | número máximo de diversificação; |
| **(ndiver2)** | número de elementos candidatos a diversificação; |
| **(nduramin)** | número mínimo de iterações tabu; |
| **(nduravar)** | número variável de iterações tabu; |
| **(nintens)** | número máximo de intensificação sem melhoria; |
| **(nmax)** | número máximo de linhas permitido; |
| **(nplmax)** | número máximo de PL's; |
| **(nsaen)** | proposta aleatória para sair; |
| **(nsem2)** | número de sementes (II etapa do processo); |
| **(nsem)** | número de sementes (I etapa do processo); |
| **(ntrocas)** | máximo número de trocas permitidas. |

# *Sumário*

# 1 INTRODUÇÃO

## 1.1 Descrição do Problema

O Problema do Planejamento da Expansão dos Sistemas de Transmissão (PPET) pode ser formulado como um problema de programação não linear inteiro misto (PNLIM) que apresenta como uma de suas características o fenômeno da explosão combinatória. Para tentar contornar esse problema os planejadores usam uma diversidade de métodos aproximados e de otimização clássica para resolvê-lo. As principais dificuldades na resolução deste problema estão relacionadas com a natureza combinatória do processo de planejamento que normalmente leva a um número explosivo de alternativas, principalmente no caso de sistemas de médio e de grande porte. Além disso, o PPET apresenta uma estrutura multimodal com um número muito elevado de ótimos locais, o que leva a maioria dos métodos aproximados a parar numa solução ótima local e às vezes de pouca qualidade. Para contornar estes problemas foram apresentadas muitas metodologias na literatura especializada.

Neste trabalho o PPET é analisado sobre o ponto de vista estático, para uma configuração inicial e os dados de geração e demanda do horizonte de planejamento (além de outros dados como limites de operação, custos e restrições de investimento), e procura-se determinar o plano de expansão com um custo mínimo, isto é, determina-se *onde* e que *tipos* de novos equipamentos devem ser instalados na rede de transmissão.

Na literatura especializada existem vários tipos de modelos ou formulações matemáticas para realizar a modelagem do PPET, os mais importantes são: modelo DC, modelo de transportes e modelo híbrido. Tradicionalmente, o modelo DC é considerado como sendo o ideal para representar o PPET, sendo os outros modelos versões relaxadas ou simplificadas do modelo DC. Para resolver estes modelos são usadas várias técnicas que podem ser agrupadas em dois grupos: *métodos analíticos e métodos aproximados.*

*Os métodos analíticos* usam técnicas de decomposição matemática e apresentam a

característica de que encontram a solução ótima do (PPET) e são muito eficientes em sistemas de pequeno e médio porte, mas para sistemas de grande porte ainda apresentam problemas de elevado esforço computacional e de convergência.

*Os métodos aproximados* se subdividem em dois grupos: os algoritmos heurísticos construtivos (AHC) e no outro grupo estão as metaheurísticas. Os AHC, apresentam a vantagem de fornecer soluções rápidas com esforços computacionais pequenos, mas os mesmos raramente encontram a solução ótima de sistemas de grande porte e não fornecem informação da qualidade da solução obtida, isto é, o quão perto da solução ótima se encontra a solução obtida. Entretanto, estes métodos ainda são os mais usados pelos setores de planejamento das empresas de energia elétrica nos trabalhos de planejamento. O grupo das metaheurísticas apresentam um conjunto de técnicas de otimização adaptadas para lidar com problemas complexos e que apresentam a característica da explosão combinatória. Dentre as várias técnicas, existem algumas mais importantes: *"Simulated Annealing"* (SA), Algoritmos Genéticos (AG), Busca Tabu (BT); estas técnicas são aplicadas com muito sucesso para resolver muitos problemas do campo da pesquisa operacional e também em alguns problemas de engenharia elétrica. Estes métodos apresentam características gerais de que convergem para soluções ótimas ou quase ótimas mas com um esforço computacional elevado.

Busca Tabu é um procedimento metaheurístico utilizado para gerenciar um algoritmo heurístico de busca local evitando que o processo pare em um ótimo local. Assim, BT realiza uma exploração através do espaço de configurações contornando adequadamente os ótimos locais. A filosofia da BT é derivar e explorar uma coleção de estratégias inteligentes para a resolução de problemas, baseados em procedimentos implícitos e explícitos de apredizagem. Está baseada na premissa de que a resolução de um problema pode ser considerado inteligente se esse processo incorpora a *memória adaptativa* e a *exploração sensível*.

Tendo a metaheurística BT como técnica de busca, o problema do processo de busca se encerrar em ótimos locais, está resolvido, mas surge um novo problema que é o tempo computacional. Este problema pode ser solucionado recorrendo-se às técnicas e ferramentas da programação paralela. Atualmente, com a disponibilidade de máquinas paralelas virtuais, com desempenho e confiabilidade crescentes e custos cada vez menores, estas técnicas vêm contribuindo sobremaneira para o sucesso da aplicação de métodos de otimização como uma solução prática e computacionalmente viável.

Levando em consideração esse problema do esforço computacional, muitos problemas

interessantes de otimização não podem ser resolvidos de forma exata, utilizando a computação convencional (serial) dentro de um tempo razoável, inviabilizando sua utilização em muitas aplicações reais na engenharia e na indústria. Embora os computadores estejam cada vez mais velozes, existem limites físicos e a velocidade dos circuitos não pode continuar melhorando indefinidamente. Por outro lado nos últimos anos tem-se observado uma crescente aceitação e uso de implementações paralelas nas aplicações de alto desempenho como também nas de propósito geral, motivadas pelo surgimento de novas arquiteturas que integram dezenas de processadores rápidos e de baixo custo. Essas arquiteturas compõem ambientes com memória distribuída como, por exemplo, estações de trabalho ou PCs conectados em rede.

A computação paralela tem como objetivo usar simultaneamente um conjunto de processadores interligados, de modo a resolver um problema conjuntamente mais rápido que o processamento seqüencial (usando somente um processador), através de uma coleção de processadores, tipicamente do mesmo tipo, interconectados de maneira a permitir a coordenação de suas atividades e a troca de dados. O uso de computadores paralelos se faz necessário em diversos problemas onde o volume de dados e cálculos é grande e precisa-se de rapidez na obtenção das respostas.

Com a computação paralela, a criação de algoritmos modifica-se sensivelmente. As estratégias utilizadas em algoritmos seqüenciais para resolver um problema não servem totalmente para a criação de algoritmos paralelos para o mesmo problema.

No processamento paralelo existem duas bibliotecas baseadas na troca de mensagens usadas em multicomputadores, são a MPI *("Message Passing Interface")* e a PVM *("Parallel Virtual Machine")*. Neste trabalho usa-se a biblioteca MPI, esta biblioteca oferece mais recursos que a biblioteca PVM, com mais opções e mais parâmetros por chamada, e vem se tornando o padrão das implementações paralelas.

A seguir é apresentada a seqüência de desenvolvimento deste trabalho.

No Capítulo **II**, é apresentado o estado da arte do PPET, sendo citadas algumas importantes referências.

No Capítulo **III**, é apresentado alguns modelos matemáticos para modelagem do PPET e a modelagem usada neste trabalho: modelo DC. São apresentadas também algumas técnicas que podem ser usadas para resolver o PPET.

No Capítulo **IV**, é apresentada a teoria básica de Busca Tabu, baseada no trabalho de (GLOVER, 1986).

No Capítulo **V**, é apresentado o algoritmo serial, com as estratégias de buscas e as técnicas que foram usadas na sua implementação.

No Capítulo **VI**, é apresentado uma descrição sobre computação paralela com MPI.

No capítulo **VII**, são apresentadas as versões paralelas desenvolvidas.

No Capítulo **VIII**, são apresentados testes e resultados com sistemas de pequeno, médio e grande porte.

No Capítulo **XIX**, são apresentadas as conclusões e sugestões para futuros trabalhos.

No final, são apresentadas as referências bibliográficas, o Apêndice A e B onde são apresentados os passos para executar programas paralelos e montar o ambiente paralelo, e também os dados dos sistemas testes.

# 2 ESTADO DA ARTE

## 2.1 Introdução

Diversas publicações descrevem bem o problema geral do PPET, veja em (PÉREZ-ARRIAGA; GÓMEZ; RAMOS, 1987), (PUNTEL et al., 1984) e (SULLIVAN, 1977).

Nos últimos anos, as pesquisas na área de modelos e técnicas de solução de planejamento da transmissão experimentou um aumento razoável. Muitos artigos e relatórios sobre novos modelos e técnicas de solução foram publicados na literatura especializada, principalmente pela melhoria do desempenho dos computadores. Assim novos algoritmos de otimização foram desenvolvidos.

O PPET é um problema complexo, de difícil resolução e 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, preestabelecidos).

O problema de planejamento estático da expansão da transmissão num horizonte de longo prazo pode ser formulado como um problema de programação não linear inteiro misto (PNLIM) e dadas as dimensões que o problema assume para casos práticos em geral observa-se o fenônemo da explosão combinatória (pertencente ao conjunto de problemas NP-completo, de difícil tratamento). Sendo que, para uma alternativa de investimento (uma dada configuração), o problema se reduz a um problema de programação linear cujo objetivo é verificar a factibilidade desta alternativa.

O problema clássico de planejamento de transmissão foi estudado em detalhe em (LATORRE, 1993), que mostra uma visão geral e uma análise qualitativa detalhada das principais publicações no planejamento estático da transmissão usando modelos de otimização.

O trabalho de (GALLEGO, 1997) apresenta detalhadamente as modelagens matemática e as técnicas aproximadas de resolução do PPET. Este trabalho baseia-se fundamentalmente nos conceitos ali apresentados. Na seqüência, são apresentadas mais informações sobre trabalhos que tratam da modelagem matemática, dos métodos aproximados e outros trabalhos relevantes na área.

## 2.2 Modelagem Matemática

Nesta seção será feito um breve relato sobre o modelo DC, usado neste trabalho, e mostrados alguns trabalhos que usam esse modelo. Além desta modelagem, existem outros modelos matemáticos importantes como: modelo de transportes, modelo híbrido, modelo disjuntivo e modelo AC. Em (ROMERO; MANTOVANI; HAFFNER, 2002) foram estudados esses modelos matemáticos, que tem sido usados na literatura para representar o estudo das redes de transmissão em planejamento da expansão da transmissão.

O modelo DC, foi o mais usado em planejamento da expansão de sistemas de transmissão. Nesse modelo todos os circuitos devem obedecer a LKT (Leis de Kirchhoff das tensões). Assim, o modelo matemático é um problema de programação não linear (PNLIM) de elevada complexidade. No Capítulo 3, na seção 3.1.1 são apresentados mais detalhes do modelo DC. Na literatura especializada existem muitos trabalhos usando este modelo, um desses trabalhos é de (ROCHA, 2004) que teve uma grande contribuição para a extensão do modelo DC. A idéia básica desse trabalho consiste em resolvê-lo, após o relaxamento da integralidade das variáveis de investimento, ou seja, $n_{ij}inteiro = n_{ij} \geqslant 0$, e o problema acaba se tornando um problema de PNL.

Um outro modelo matemático muito usado para o problema do planejamento da transmissão é o modelo de transportes, que iniciou-se com o trabalho de Garver (GARVER, 1970) e representou uma proposta fundamental na pesquisa em planejamento da expansão de sistemas de transmissão. O modelo de transportes leva em conta apenas a LKC (Lei de Kirchhoff das correntes) e a capacidade de operação de circuitos e geradores. Portanto, não é levado em conta a LKT (Leis de Kirchhoff das tensões). Na literatura existem alguns trabalhos usando o modelo de transportes, como no trabalho de (ROMERO et al., 2001).

Já o modelo híbrido combina características do modelo DC e do modelo de transportes. No modelo de híbrido apenas uma parcela dos circuitos são obrigados a obedecer a LKT. A idéia de usar este tipo de modelo é tentar encontrar soluções ótimas do modelo DC,

mas sem adicionar a complexidade do problema. Em (ROMERO; MANTOVANI; HAFFNER, 2002), pode se visto com detalhes a formulação deste modelo.

O modelo híbrido é dividido em duas formas: o modelo híbrido linear e o modelo híbrido não linear. O modelo híbrido linear é mais simples, pois os circuitos que já existem na topologia base são obrigados a obedecer a LKT. Em (ROCHA, 2004) foi usado o modelo híbrido linear no PPET. Outro trabalho que se destacou usando o modelo híbrido foi o trabalho de (ROMERO et al., 2007), onde são feitas aplicações para o PPET usando o modelo híbrido linear.

## 2.3 Métodos Aproximados

As técnicas de otimização que podem ser usadas na resolução do PPET são classificadas em: (1) *métodos analíticos* e (2) *métodos aproximados*. Os métodos analíticos se baseam na decomposição matemática. Alguns exemplos desse método são: o algoritmo de branch-and-bound e o método de decomposição matemática (decomposição de Benders). Os métodos aproximados se dividem em duas classes: os *algoritmos heurísticos construtivos* e as *metaheurísticas*. Nesta seção será tratada particularmente os métodos aproximados.

Dentre os algoritmos heurísticos construtivos que existem na literatura para se resolver o PPET, os mais usados são: o algoritmo de Garver, o algoritmo de Villasana-Garver-Salon (VGS), que é uma extensão do algoritmo de Garver e os algoritmos de Mínimo Esforço e Mínimo Corte de Carga.

Dentre as metaheurísticas existentes as mais importantes e as mais usadas na resolução do PPET são: Algoritmos Genéticos, Busca Tabu e *"Simulated Annealing"*. Neste trabalho é usada a técnica Busca Tabu, que será detalhada posteriormente.

### 2.3.1 Algoritmos Heurísticos Construtivos

Na literatura encontram-se muitos trabalhos sobre os algoritmos heurísticos construtivos e na maioria desses trabalhos estes algoritmos nem sempre encontram a solução ótima da expansão de um sistema elétrico. (SILVA; AREIZA; GIL, 2000), trabalha com o planejamento da transmissão usando métodos heurísticos.

Na prática esses algoritmos encontram as configurações ótimas de sistemas pequenos e apenas configurações boas para sistemas elétricos de médio e grande porte. Entretanto,

estes algoritmos são muito importantes pelos seguintes motivos:

- Na primeira fase de pesquisa (década de 60 e 70), esta era a única ferramenta que existia para solucionar os problemas de planejamento de sistemas elétricos de grande porte;
- A maioria destes algoritmos são robustos e simples de entender, programar e usar;
- O esforço computacional destes algoritmos é muito pequeno;
- Muitas características e propriedades destes algoritmos podem ser usadas no desenvolvimento de algoritmos mais complexos como as metaheurísticas (SA, AG, BT, GRASP, etc...);
- Ainda hoje esses algoritmos são muito usados pelas empresas de energia elétrica.

Os métodos de solução que utilizam estas técnicas fazem o plano de expansão através de um processo passo-a-passo em que, para uma dada configuração denominada configuração base ou inicial, os equipamentos que aumentam a capacidade do sistema conforme o aumento da demanda, são adicionados um a um ou em pequenos grupos. Assim, a configuração do sistema é modificada pela adição de um ou vários circuitos e a configuração então obtida é denominada de configuração corrente.

A desvantagem deste tipo de método é que este procedimento não garante a otimalidade de uma solução. Os métodos que adotam técnicas heurísticas não conseguem sequer avaliar a qualidade das soluções que encontram.

A grande vantagem deste tipo de método de solução é a simplicidade da formulação. Em alguns casos pode-se dizer que o método é uma tentativa do planejador de transformar seu conhecimento em um programa, adicionando alguma ferramenta matemática e contando com a lógica humana para a verificação da validade dos resultados.

Dentre os algoritmos heurísticos usados no planejamento de sistemas de transmissão, o mais importante deles é o algoritmo de Garver. Além de ter sido um dos primeiros algoritmos apresentados em planejamento, a idéia apresentada por Garver ainda é de grande valor. Na literatura, um trabalho que contribuiu para a pesquisa foi (ROCHA et al., 2003), onde é analisado a aplicação de algoritmos para o modelo de transportes no PPET.

Em (ROCHA, 2004) pode ser visto o algoritmo de Villasana-Garver-Salon, para o PPET. Uma característica especial do algoritmo de VGS é que a estratégia de otimização usa o modelo híbrido linear no processo de otimização.

Existem outros dois algoritmos heurísticos que são importantes na área de pesquisa do PPET, o algoritmo de mínimo esforço, que é apresentado com detalhes no trabalho (MONTICELLI et al., 1982) e o algoritmo de mínimo Corte de Carga apresentado em (PEREIRA; PINTO, 1985).

### 2.3.2 Metaheurísticas

As metaheurísticas são técnicas de busca que combinam métodos heurísticos. Estas técnicas têm se mostrado muito efetivas nas soluções de problemas complexos de grande porte e que apresentam a característica da explosão combinatória, ou seja, ocorre um aumento computacional conforme aumenta a diminuição do espaço de busca do problema. Um exemplo de problema complexo é o PPET. Em (ROMERO; MANTOVANI, 2004), pode ser visto uma introdução sobre as metaheurísticas.

Dentre as metaheurísticas existem as técnicas de *("Simulated Annealing")* , Algoritmos Genéticos, GRASP, Busca Tabu *("Tabu Search")*, entre outras. A seguir será mostrada uma dedução breve de algumas destas técnicas e alguns tabalhos desenvolvidos nessa de linha de pesquisa. Neste trabalho usa-se a metaheurística Busca Tabu, que será melhor detalhada no Capítulo 4.

#### 2.3.2.1 Simulated Annealling

Um dos primeiros trabalhos desenvolvido de *("Simulated Annealing"*, SA), foi para analisar o comportamento microscópico dos corpos. Um trabalho da mecânica estática usando a metodologia de Monte Carlo, que ficou conhecido como o algoritmo de Metrópolis, e foi usado pelos físicos para desenvolver uma técnica de construção de cristais perfeitos. Uma análise completa de SA pode ser vista em (ROMERO; MANTOVANI, 2004) e no trabalho de (GALLEGO, 1997).

Na literatura existem vários trabalhos com SA na área do PPET. Um dos trabalhos pioneiros encontra-se em (GALLEGO; ROMERO; MONTICELLI, 1996) onde a rede é modelada usando o modelo DC. Outros trabalhos foram desenvolvidos com SA usando a programação paralela, como os trabalhos de (GALLEGO et al., 1997) e (DEOLIVEIRA, 2004), que usaram a biblioteca para troca de mensagens PVM, nestes trabalhos foram de-

senvolvidos vários algoritmos paralelos, voltados para o planejamento estático de sistemas e transmissão.

#### 2.3.2.2 Algoritmo Genético

O algoritmo Genético é uma metodologia usada para resolver problemas de otimização combinatória e alcançou um grande sucesso na última década para solucionar problemas de grande porte em áreas muito diversas. O algoritmo foi, inicialmente, formulado por Holland (GOLDBERG, 1989), baseado no princípio da seleção natural que acontece na natureza e que fornece maiores chances de sobrevivência aos indivíduos melhores adaptados ao meio ambiente. De acordo com (ROCHA, 2004), matematicamente o algoritmo pode ser considerado como uma técnica de otimização combinatória com uma alta probabilidade de encontrar a solução ótima global de problemas grandes e complexos e com muitas soluções ótimas locais.

(GALLEGO, 1997) e (DEOLIVEIRA, 2004) usaram AG no planejamento estático, sendo que (DEOLIVEIRA, 2004) criou diversos algoritmos paralelos.

#### 2.3.2.3 Busca Tabu

A BT é um técnica metaheurística que é muito utilizada na resolução de problemas complexos como o PPET. BT foi desenvolvida a partir de conceitos usados na inteligência artificial e, diferentemente de outras técnicas, não teve sua origem relacionada a processos de otimização biológicos ou físicos. Foi proposta por Fred Glover na década de 80 e está sendo intensamente utilizada para resolver problemas complexos em diversas áreas de pesquisa operacional. Em (ROMERO; MANTOVANI, 2004) é feito um resumo de toda a teoria de BT e é mostrado o desempenho do algoritmo BT, usando o problema das $n$ rainhas.

Em (GALLEGO; ROMERO; MONTICELLI, 2000) e (SILVA et al., 2001) são apresentados algoritmos com técnica de solução baseada em BT, e utilizam o modelo DC, como modelagem matemática do problema.

Em (GALLEGO; ROMERO; MONTICELLI, 2000) o algoritmo é dividido em três fases distintas: durante a primeira fase, o algoritmo BT é introduzido basicamente como um algoritmo de busca com uma memória de curto prazo, uma lista tabu e um critério de aspiração que permite determinar dentre os movimentos permitidos, aqueles mais atrativos; na segunda fase é incluído um mecanismo de diversificação, um de intensificação

e um de memória a longo prazo; e finalmente, na terceira fase, são incluídos *"Path Relinking"*, configurações de elite, seleção inteligente de configuração inicial, oscilação estratégica, redução de vizinhos e versões híbridas adicionando características de outros métodos combinatóriais como Algoritmo Genético e *"Simulated Annealing"*.

O algoritmo de (GALLEGO; ROMERO; MONTICELLI, 2000) foi testado com alguns sistemas teste, no sistema de 6 barras de Garver, no sistema de 46 barras Sul brasileiro e no sistema de 87 barras Norte-Nordeste brasileiro.

Para o sistema de 6 barras de Garver, o algoritmo iniciou, com configurações iniciais geradas aleatoriamente pois da outra forma a busca se tornaria trivial. Para esse teste não foi permitido o redespacho da geração, tornando o problema mais difícil e sua convergência mais complicada. O algoritmo encontrou a solução ótima de $v = 200$ u.m..

Já para o sistema de 46 barras Sul brasileiro, as configurações iniciais foram geradas aleatoriamente e pelo método de Garver. A configuração ótima para este caso teve um custo de investimento de $v = 154{,}420$ milhões de doláres.

Para o sistema Norte-Nordeste brasileiro de 87 barras plano 2008, o algoritmo encontrou um custo de investimento de v = 2,574 bilhões de doláres com a adição de 107 circuitos.

Em (SILVA et al., 2001) o algoritmo descrito foi dividido nas seguintes etapas: na primeira etapa é obtida uma configuração inicial determinada por um conjunto de indicadores de sensibilidade; na segunda etapa, com a configuração inicial determinada, é iniciada a fase de expansão; na terceira etapa, com o término da expansão, a fase de intensificação é iniciada. Esta etapa é subdividida em duas fases: intensificação 1 e intensificação 2 e na quarta etapa, com o término da fase de intensificação, é iniciada a fase de diversificação.

Na fase de intensificação, "um movimento" consiste em trocar dois ou mais circuitos, ou seja, um circuito candidato construído previamente é removido e outro candidato é adicionado. Este movimento (permuta) é baseado em indicadores e são permitidos somente dentro de uma região factível. Na intensificação 1, apenas movimentos para soluções com menores custos de investimentos são permitidos e em contrapartida, na intensificação 2, movimentos para soluções mais caras são permitidos com o objetivo de contornar problemas com soluções ótimas locais. Na fase de diversificação, o objetivo é direcionar a busca para regiões não exploradas dentro do espaço de buscas. Nesta fase, os circuitos candidatos que apresentam uma maior freqüência na última iteração serão proibidos na próxima fase de expansão. O critério de parada do algoritmo é o número de

fases de diversificação executada.

O algoritmo de (SILVA et al., 2001) foi testado usando os sistemas teste de 46 barras Sul brasileiro e o sistema de 79 barras Sudeste brasileiro. Para o sistema de 46 barras Sul brasileiro, encontrou-se a solução ótima com um custo de investimento de $v = 154{,}420$ milhões de doláres. No sistema Sudeste brasileiro de 79 barras a melhor solução encontrada pelo algoritmo teve um custo de investimento de $v = 444{,}390$ milhões de doláres, com a adição de 21 circuitos.

### 2.3.3 Outros Trabalhos Relevantes

Dentre os trabalhos e artigos publicados cita-se alguns, que serão o alicerce de pesquisa para este projeto, dando ênfase para os trabalhos voltados na área de planejamento estático de sistemas de transmissão usando o algoritmo de BT e da programação paralela no ambiente MPI. A metodologia tabu, as técnicas da BT e a implementação da paralelização deste algoritmo no ambiente MPI, serão o tema principal deste trabalho. Em (MONTICELLI; SHIOZER; SOUZA, 1998), é apresentado um relatório de como a paralelização pode ajudar a resolver problemas complexos de otimização.

O trabalho inicial de pesquisa, foi de (GALLEGO, 1997), que contribuiu em grande parte para o desenvolvimento de bons algoritmos para a resolução do PPET, usando a BT como técnica de busca e aproveitando bem suas técnicas e estratégias de busca. A partir deste trabalho, foram desenvolvidos diferentes algoritmos paralelos para o PPET (DEOLIVEIRA et al., 2001) e (DEOLIVEIRA, 2004).

Foram também pesquisadas outras publicações importantes sobre a metaheurística BT, dentre estas estão alguns trabalhos: (GLOVER,1989a) e (GLOVER,1989b), (GLOVER; LAGUNA, 1993), (GLOVER, 1986), (LAGUNA, 1995), (GLOVER; LAGUNA, 1997); e (WEN; CHANG, 1997) que usou um algoritmo BT para a resolução do planejamento da expansão da transmissão.

Para o desenvolvimente deste trabalho, pequisou-se alguns trabalhos importantes voltados para a programação paralela. (BARNEY, 2002), faz uma introdução completa sobre a programação paralela em (MPIFORUM, 1995), vê-se um amplo conteúdo sobre a biblioteca para troca de mensagens MPI. Outro trabalho usado como uma das fontes para pesquisa foi de (DEOLIVEIRA, 2004), onde trabalha-se com programação paralela com diferentes algoritmos e diferentes metaheurísticas.

# 3 MODELAGENS E TÉCNICAS USADAS NA RESOLUÇÃO DO PPET

Neste capítulo serão apresentados os modelos matemáticos de uma forma geral, mas inicialmente será dada mais ênfase ao modelo que foi utilizado neste trabalho e também será feita uma análise geral sobre várias metodologias que podem ser usadas na resolução do problema de planejamento estático da expansão da transmissão de energia elétrica.

## 3.1 Modelagem do Problema

### 3.1.1 Modelo DC

Neste trabalho, usa-se o modelo DC, como modelagem para o problema do planejamento. O modelo DC é considerado ideal, pois leva em conta as duas leis de Kirchhoff para o sistema elétrico e a capacidade de transmissão das linhas existentes e das candidatas.

Esta formulação modificada, é dada em (3.1-3.7):

$$\min v = \sum_{(i,j)\in\Omega} c_{ij} n_{ij} + \sum_{i} \alpha_i r_i \tag{3.1}$$

s.a.

$$Sf + g + r = d \tag{3.2}$$

$$f_{ij}^0 - \gamma_{ij} n_{ij}^0 (\theta_i - \theta_j) = 0 \tag{3.3}$$

$$|f_{ij}| \leq (n_{ij}^0 + n_{ij})\overline{f}_{ij} \tag{3.4}$$

$$0 \leq g \leq \overline{g} \tag{3.5}$$

$$0 \leq n_{ij} \leq \overline{n}_{ij} \tag{3.6}$$

$$0 \leq r \leq d \tag{3.7}$$

$n_{ij}$ inteiro, $f_{ij}$ irrestrito, $\theta_j$ irrestrito

$(i,j) \in \Omega$

A restrição (3.2) representa a LKC e a restrição (3.3), a LKT.

O modelo apresentado em (3.1-3.7) está modificado em relação ao modelo DC original. Uma das modificações feitas é que foi acrescentado o segundo termo $\sum_i \alpha_i r_i$ na função objetivo para facilitar o processo de resolução. Por exemplo, se o valor de $\alpha$ é relativamente grande, então na solução final todos os valores de $r_i$ deverão ser iguais a zero se o problema for factível. E para esta condição, a solução obtida é exatamente igual para a formulação original e para a formulação modificada.

### 3.1.2 Modelo de Transportes

O modelo de transportes foi formulado por Garver (GARVER, 1970), sendo uma das primeiras propostas para planejamento de redes de transmissão que usou programação linear. Esta metodologia consiste basicamente em resolver de maneira aproximada uma versão relaxada do modelo DC. No modelo de Garver, conhecido como *modelo de transportes*, leva-se em conta a lei de Kirchhoff das correntes. Ou seja, não se leva em conta o conjunto de restrições correspondentes a lei de Kirchhoff das tensões.

Assim, o modelo de transportes assume a formulação apresentada em (3.8-3.13), em que $S$ é a matriz de incidência nó-ramo e $f_{ij}$ o fluxo de potência no ramo $ij$. O conjunto de restrições $Sf + g = d$ representa as equações correspondentes a primeira lei de Kirchhoff; as restrições $|f_{ij}| \leq (n_{ij}^0 + n_{ij})\overline{f}_{ij}$ representam a capacidade de transmissão dos circuitos

(linhas e/ou transformadores) e as demais restrições de limites de geração, de cargas e de circuitos adicionados em cada ramo $ij$.

Levando em conta as observações anteriores, o modelo de transportes para o problema de planejamento de sistemas de transmissão pode ser formulado por (3.8-3.13), como segue.

$$\min v = \sum_{(i,j)\ \in\ \Omega} c_{ij} n_{ij} \tag{3.8}$$

s.a.

$$Sf + g = d \tag{3.9}$$

$$|f_{ij}| \leq (n^0_{ij} + n_{ij})\overline{f}_{ij} \tag{3.10}$$

$$0 \leq g \leq \overline{g} \tag{3.11}$$

$$0 \leq n_{ij} \leq \overline{n}_{ij} \tag{3.12}$$

$$f_{ij} \text{ irrestrito, } n_{ij} \text{ inteiro} \tag{3.13}$$

### 3.1.3 Modelo Híbrido

Este modelo é uma combinação do modelo de transportes e o modelo DC. Neste modelo, trata-se de contornar as desvantagens na solução do modelo de transportes, adicionando-se somente uma parcela de restrições correspondentes a segunda lei de Kirchhoff, isto é, são consideradas como parte da formulação do problema aquelas restrições da segunda lei de Kirchhoff correspondentes às linhas existentes e eliminadas as restrições correspondentes aos novos caminhos. Logo, no modelo híbrido, deve-se satisfazer a primeira lei de Kirchhoff em todas as barras do sistema e a segunda lei de Kirchhoff somente nos laços já existentes na configuração base.

A formulação, assim definida, foi introduzida por Garver em (VILLASANA; GARVER; SALON, 1985). A formulação (3.14-3.21) apresenta a formulação correspondente a este modelo. O conjunto de restrições $Sf + S^0 f^0$ representa as equações da primeira e da segunda lei de Kirchhoff para todas as barras do sistema e para os laços existentes na topologia base, em que $S$ é a matriz de incidência nó-ramo dos circuitos que não têm linhas na configuração base.

Levando em conta estas observações o modelo híbrido pode ser formulado por (3.14-3.21), como segue.

$$\min v = \sum_{(i,j)\in\Omega} c_{ij} n_{ij} \tag{3.14}$$

s.a.

$$Sf + S^0 f^0 + g = d \tag{3.15}$$

$$f_{ij}^0 - \gamma_{ij} n_{ij}^0 (\theta_i - \theta_j) = 0,\ \forall (i,j) \in \Omega_0 \tag{3.16}$$

$$|f_{ij}^o| \leq n_{ij}^0 \overline{f}_{ij},\ \forall (i,j) \in \Omega_0 \tag{3.17}$$

$$|f_{ij}| \leq n_{ij} \overline{f}_{ij},\ \forall (i,j) \in \Omega \tag{3.18}$$

$$0 \leq g \leq \overline{g} \tag{3.19}$$

$$0 \leq n_{ij} \leq (\overline{n}_{ij} - n_{ij}^1) \tag{3.20}$$

$$f_{ij}^0,\ \text{e}\ f_{ij}\ \text{irrestritos},\ \theta_j\ \text{irrestrito},\ (i,j) \in \Omega \tag{3.21}$$

Descrição das variáveis das formulações apresentadas:

$v$: é o investimento devido as adições de circuitos.

$c_{ij}$: é o custo de um circuito que pode ser adicionado no caminho $i-j$.

$n_{ij}$: é o número de circuitos adicionados no processo de otimização.

$n_{ij}^0$: é o número de circuitos existentes na topologia base.

$\overline{n}_{ij}$ é o número máximo de circuitos que podem ser adicionados no caminho $i-j$.

$S$: é a matriz de incidência nó-ramo transposta do sistema elétrico.

$S^0$: é a matriz de incidência nó-ramo transposta para os circuitos da topologia base.

$f$, é o vetor de fluxos com elementos $f_{ij}$ (o fluxo de potência através dos circuitos).

$f^0$ é o vetor de fluxos através dos circuitos da topologia base, com elementos $f_{ij}^0$.

$\overline{f}_{ij}$: é a capacidade de transmissão de um circuito no caminho $i-j$.

$r$: é o vetor de geradores fictícios ou artificiais.

$g$: é o vetor geração com elementos $g_k$ (geração na barra $k$).

$\overline{g}$: é a geração máxima.

$d$: é o vetor de cargas.

$\Omega$ é o conjunto de índice dos circuitos candidatos.

$\Omega_0$ é o conjunto de índice dos circuitos presentes na topologia base.

## 3.2 Técnicas Usadas para Resolver o Problema de Planejamento

### 3.2.1 Métodos Analíticos

Na década de 80, iniciou-se uma nova fase na tentativa de resolver a formulação (3.1-3.7) de maneira ótima e a principal ferramenta matemática encontrada foram as técnicas de decomposição matemática. O objetivo principal era o de encontrar a solução ótima do PPET usando o modelo DC, mas, para sistemas de grande porte ainda apresentam problemas de esforço computacional e de convergência para resolver a formulação (3.1-3.7), o que significa resolver um problema de PNLIM.

Nesta perspectiva, a metodologia mais usada foi a técnica de decomposição de Benders a qual explora a decomposição natural do PPET em duas partes, ou seja:

- **Um subproblema de investimento** em que se escolhe um plano de expansão candidato e, calculam-se os custos de investimento associados ao mesmo;
- **Um subproblema de operação** onde é testado o plano de expansão candidato em termos do adequado atendimento da carga.

A otimização global é atingida através de uma resolução iterativa dos subproblemas de operação e investimento.

Como as metodologias de decomposição foram incapazes de resolver sistemas de grande porte como o sistema Norte-Nordeste, foram iniciadas novas pesquisas relacionadas com os métodos de otimização combinatórias, cujas características fundamentais são as de resolver sistemas de grande porte, chegar a soluções próximas ao ótimo global e obter soluções em tempos de computação razoáveis.

### 3.2.2 Métodos Aproximados

Na literatura especializada são apresentados diversos métodos aproximados que determinam planos da expansão de redes de transmissão a longo prazo. Um dos primeiros métodos aproximados desenvolvidos foi o de Garver (GARVER, 1970); em que se propõe

um algoritmo heurístico construtivo para encontrar uma boa configuração e não necessariamente a configuração ótima. Estes métodos geralmente são de fácil implementação e requerem pouco esforço computacional; com relação a qualidade da resposta, e para sistema de médio e grande porte, seu valor fica geralmente afastado da resposta ótima. Além do método de Garver, menciona-se o método de Villasana (VILLASANA; GARVER; SALON, 1985), que é uma extensão do método anterior. Sendo que o mesmo trabalha usando o modelo híbrido. Também cabe destacar o *método do mínimo esforço* usando o modelo DC (MONTICELLI et al., 1982) e o *método de mínimo corte de carga*, baseado em um problema linear especializado obtido do modelo DC (PEREIRA; PINTO, 1985); e estas metodologias ainda são empregadas no planejamento de sistemas de transmissão nas empresas de energia elétrica.

#### 3.2.2.1 Algoritmos Heurísticos Construtivos

**Metodologia de Garver**

O algoritmo de Garver (GARVER, 1970) 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. Esta metodologia considera dois tipos de ligações:

- *Ligações Normais:* Com capacidade de transmissão máximas iguais às capacidades das 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.

Na metodologia de Garver, todo fluxo que não puder ser transportado pelas ligações normais, fluirão pelas ligações fictícias com custos elevados, pois estas têm capacidades ilimitadas, e só passarão através das ligações de fictícias quando for impossível transportá-los pelas ligações normais, já que estas tem custos muito inferiores. Em cada estágio do processo de planejamento, deve-se resolver um problema de programação linear e assim 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. A maior limitação da metodologia é que ela não garante a obtenção da solução ótima

do sistema planejado. Portanto, a metodologia de Garver, em essência, é de naturaleza heurística.

A estrutura básica do algoritmo de Garver é:

1. Tomar a configuração base como configuração corrente.
2. Resolver um PPET de fluxo de rede com custo mínimo para a configuração corrente. Se não existirem mais novos caminhos a serem inseridos, então pare. Caso contrário, ir para o passo 3.
3. Calcular os fluxos através de todos os novos circuitos adicionados pelo PPET. Atualizar a configuração corrente adicionando um circuito para o novo caminho que apresentar o maior valor de fluxo. Voltar ao passo 2.

A metodologia proposta por Villasana é uma extensão da metodologia de Garver, na qual se adiciona a segunda lei de Kirchhoff para a rede existente. Com isto, gera-se a formulação apresentada como modelo híbrido que é resolvida usando uma metodologia muito parecida com a metodologia de Garver. Aqui também não se garante a otimalidade da solução obtida. Nesta metodologia, mantêm-se os conceitos de linhas de sobrecarga e, usa-se programação linear para determinar o circuito mais sobrecarregado e, portanto, candidato à adição de um novo circuito.

A estrutura básica do algoritmo de Villasana-Garver é dada por:

**Fase I:**

1. Tomar a configuração base como configuração corrente.
2. Resolver um PPET para a configuração corrente. Se não existirem mais novos caminhos a serem inseridos, então pare. Caso contrário, ir para o passo 3.
3. Calcular os fluxos através de todos os novos circuitos adicionados pelo PPET. Atualizar a configuração corrente adicionando um circuito para o novo caminho que apresentar o maior valor de fluxo. Voltar ao passo 2.

**Fase II:**

1. Ordenar os circuitos adicionados em ordem decrescente de seus custos e procurar eliminar aqueles cuja saída não produzem cortes de carga no sistema;

2. FIM.

**Metodologia do Mínimo Corte de Carga**

O algoritmo de Mínimo Corte de Carga (MCC) (PEREIRA; PINTO, 1985), é um algoritmo heurístico de tipo construtivo que em cada passo do algoritmo produz a adição de um circuito na configuração base. Este algoritmo também é conhecido como 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.

Em cada iteração deste algoritmo é realizada a adição de um circuito e esse circuito é selecionado de acordo a um índice de desempenho ou índice de sensibilidade **(IS)**.

A estrutura básica do algoritmo de Mínimo Corte de Carga é dada por:

**Fase I:**

1. Tomar a Configuração base como configuração corrente.
2. Resolver um PPET para corte mínimo de carga para a configuração corrente. Se não existirem sobrecargas, então ir para a Fase II. Caso contrário, calcular os $IS_{mcc}$ e ordenar os circuitos candidatos iniciando pelo circuito com maior valor absoluto do índice. Ir para o passo 3.
3. Adicionar à configuração corrente o primeiro circuito da lista anterior. Voltar ao passo 2.

**Fase II:**

1. Ordenar os circuitos adicionados em ordem decrescente de seus custos e eliminar aqueles, cuja saída, não produzem cortes de carga no sistema;
2. FIM.

O **(IS)** que permite encontrar o circuito mais atrativo para adição é determinado pela seguinte relação:

$$IS_{mcc} = (\pi_i - \pi_j(\theta_i - \theta_j)) \tag{3.22}$$

em que $\pi_j$ é o multiplicador de Lagrange da *j-ésima* restrição de igualdade $B\theta + g + r = d$ e os $\theta_j$ são os ângulos de tensão de barra obtidos ao resolver (3.22) para a configuração corrente usando um algoritmo de PL.

O **(IS)** 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 **(IS)** deve ser adicionado à configuração base, e é o melhor candidato para produzir uma maior resolução no corte de carga do sistema.

### Metodologia do Mínimo Esforço

Esta metodologia é formulada usando o modelo DC, foi criada na Unicamp por (MONTICELLI et al., 1982). Ela também 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. O critério para a adição do próximo circuito é determinado por uma análise de sensibilidade, chamada mínimo esforço.

Esta metodologia apresenta uma grande dificuldade de desconexões na rede inicial e é contornada adotando-se, superposta à configuração do sistema , uma "rede fictícia" constituída por ligações com susceptâncias iguais a, por exemplo, $10^{-4}$ vezes os valores nominais, colocada em todos os ramos onde são permitidas a construção de novas linhas. A baixa capacidade de transmissão da rede fictícia faz com que estas só sejam utilizadas quando não houver possibilidade de transporte de potência pela rede real.

Neste método emprega-se a relação seguinte, a qual é utilizada como um critério de desempenho na adição de novos circuitos ao sistema:

$$IS_{me} = \Delta Z_{ij} = -\frac{1}{2}(\theta_i - \theta_j)^2 \Delta\gamma_{ij}, \tag{3.23}$$

em que $\Delta\gamma_{ij}$ é a variação da susceptância de um circuito no ramo *ij*.

Em cada passo do processo de planejamento é adicionado ao sistema aquele circuito que produza o maior impacto na distribuição de fluxos na rede, isto é, aquele que apresenta o maior valor de $|\Delta Z_{ij}|$.

Esta metodologia tem como vantagem a sua rapidez e geralmente apresenta soluções de boa qualidade. A desvantagem é que ela não garante a otimalidade da solução e, às vezes, pode apresentar soluções distantes do ótimo global, pois não existe forma de

determinar o quanto perto a solução obtida está.

A estrutura básica do algoritmo de Mínimo Esforço é:

**Fase I:**

1. Tomar a configuração base como configuração corrente.
2. Resolver uma análise DC para a configuração corrente. Se não existirem sobrecargas, então ir para a Fase II. Caso contrário, calcular os $IS_{me}$ e ordenar os circuitos candidatos iniciando pelo circuito que apresentar maior valor absoluto do índice. Ir para o passo 3.
3. Adicionar à configuração corrente o primeiro circuito da lista anterior. Voltar ao passo 2.

**Fase II:**

1. Ordenar os circuitos adicionados em ordem decrescente de seus custos e eliminar aqueles cuja saída não produzem cortes de carga no sistema;
2. FIM.

#### 3.2.2.2 Metaheurísticas

As metaheurísticas são um conjunto de técnicas de otimização desenvolvidas para resolver problemas complexos que apresentam o chamado fenômeno da explosão combinatória.

De acordo com (ROMERO; MANTOVANI, 2004):

> *A idéia fundamental de uma metaheurística consiste em analisar ou visitar apenas um conjunto reduzido do espaço de busca, considerando que o espaço de busca é absurdamente grande.*

Uma metaheurística pode ser vista como uma estrutura algorítmica geral que possa ser aplicada aos diferentes problemas de otimização com relativamente poucas modificações para se adaptadar a um problema específico.

Algoritmos tais como: *"Simulated Annealing"*, Algoritmos Genéticos, Busca Tabu e GRASP, são algoritmos gerais de alta qualidade. Estes algoritmos são chamados de metaheurísticas, e estão sendo empregados na solução de sistemas de pequeno, médio

e de grande porte; nestes, reportam-se resultados bastante satisfatórios na maioria das aplicações apresentadas.

***"Simulated Annealing"*** é uma das técnicas usadas pelos físicos na construção de cristais perfeitos. Nesta técnica um material é aquecido até uma temperatura elevada e depois esfriado lentamente, mantendo durante o processo o chamado quase equilíbrio termodinâmico. O processo pára quando o material atinge seu estado de energia mínima na qual se transforma num cristal perfeito. Assim, o algoritmo SA tenta simular um processo equivalente para encontrar a configuração ótima de um problema complexo.

A idéia original que deu lugar a esta metaheurística é chamada de *algoritmo de Metrópolis*, o que por sua vez está baseado no *método de Monte-Carlo*, com o qual se estudam as propriedades de equilíbrio na análise do comportamento microscópico dos corpos.

O algoritmo de Metrópolis gera uma seqüência de estados de um sólido, ou seja: dado um sólido em um estado $i$ e com energia $E_i$, gera-se o estado seguinte $j$ mediante a aplicação de um mecanismo que transforma para o estado seguinte através de um pequeno distúrbio. A energia do próximo estado é $E_j$; se a diferença de energia $E_i$ - $E_j$ é menor ou igual a zero, o estado $j$ é aceito. Se a diferença de energia é maior que zero, o estado $j$ é aceito com certa probabilidade, a qual é dada por:

$$e^{\{\frac{E_i - E_j}{K_b T}\}}, \tag{3.24}$$

em que $T$ denota a temperatura, e $K_b$ é uma constante física conhecida como constante de Boltzmann. A regra de aceitação descrita é chamada *critério de Metrópolis* e o algoritmo como *algoritmo de Metrópolis*.

O **Algoritmo Genético** está baseado no princípio de seleção natural. Na natureza os indivíduos melhores dotados têm maiores chances de sobrevivência e a capacidade de adaptação a um meio mutante é fundamental na sobrevivência de indivíduos e espécies. Foi baseada nesta lei natural que as espécies sobreviveram e evoluíram na Terra. As características específicas de um indivíduo determinam sua capacidade de sobrevivência e, em última instância, essa capacidade específica é determinada pelo conteúdo genético do indivíduo, isto é, pela unidade elementar chamada *gene* na biologia. Assim, nas mudanças do material genético das espécies acontece a evolução das mesmas. A seleção natural leva à sobrevivência dos indivíduos melhores dotados e no processo de recombinação estes indivíduos transmitem aos descendentes os melhores genes; por outro lado, os indivíduos

menos dotados morrem no processo de competição por espaço, alimentos, etc. Assim, o princípio de seleção natural permite que somente os indivíduos melhores dotados gerem descendentes. Foi assim que aconteceu a evolução das espécies. A diversidade genética acontece na recombinação. Neste processo acontece uma troca de material genético que pode levar à geração de um indivíduo muito bem dotado ao receber o melhor material genético dos pais. Este processo repetido leva à evolução, isto é, uma seleção com a sobrevivência dos indivíduos melhores dotados e a correspondente reprodução (com troca de material genético) entre os mesmos. Adicionalmente, no processo de evolução existe um fenômeno chamado *mutação*, que é uma mudança do código genético dos indivíduos como conseqüência de sua interação com o meio ambiente.

No algoritmo genético o processo de otimização se inicia com a geração de uma população, isto é, um conjunto de soluções (configurações) candidatas. Cada configuração é qualificada pelo valor da função objetivo que apresenta. Geralmente a codificação de uma configuração é realizada em sistema binário. Todos os elementos da população são classificados pela qualidade de sua correspondente função objetivo. Assim, cada elemento tem uma probabilidade de passar seus genes para a geração seguinte. Os elementos melhores qualificados ("higher fitness") neste processo têm, no sentido probabilístico, maior probabilidade de participar na geração dos elementos da nova população. Essa nova geração é obtida com as operações de recombinação (*"crossover"*) e mutação.

**Busca Tabu** é um procedimento de otimização local que admite soluções de pior qualidade para escapar de ótimos locais. Em sua forma original, a cada iteração procura-se um ótimo local selecionando-se o melhor vizinho $s'$ da vizinhança $N(s)$ da solução corrente $s$. Independentemente de $f(s')$ ser melhor ou pior que $f(s)$, $s'$ será sempre a nova solução corrente. Entretanto, apenas esse mecanismo não é suficiente para escapar de ótimos locais, uma vez que pode haver retorno a uma solução previamente gerada. Para evitar isso, o algoritmo usa o conceito de lista tabu. Esta lista define todos os movimentos com um certo atributo como sendo tabu por um determinado número de iterações, conhecido como tempo tabu. Tais movimentos são proibidos a menos que a solução satisfaça a um certo critério de aspiração, em geral que essa solução seja melhor que a melhor solução encontrada até então. Os atributos são escolhidos para prevenir o retorno às soluções visitadas recentemente e por possuírem características fáceis de detectar. O procedimento chega ao fim quando alcança um certo critério de parada, geralmente um determinado número de iterações sem melhorias.

# 4 BUSCA TABU

## 4.1 Busca Tabu

Busca Tabu (BT) é um algoritmo que está sendo usado no campo da pesquisa operacional e existem muitas publicações de caráter geral, entre as quais podem ser citadas (GLOVER, 1986) e (LAGUNA, 1995).

A Busca Tabu foi introduzida por Glover em 1986 (GLOVER, 1986), que também foi quem criou o termo metaheurística. Nesta seção são apresentados a teoria e os princípios da Busca Tabu, assim como também descrevem-se seus componentes.

Busca Tabu é uma metaheurística que tem sido aplicada com sucesso em diversos problemas de otimização combinatória (GLOVER; LAGUNA, 1997). O algoritmo usa exploração sensível e memória adaptativa para guiar um procedimento de busca em vizinhança no processo de solução. Através da exploração sensível, determina-se uma direção de busca baseada em propriedades da solução corrente e da história da busca. A memória adaptativa consiste de estruturas de memória de curto e longo prazo que armazenam a história da busca. A memória de curto prazo armazena atributos de soluções já visitadas no passado recente. Estes atributos são armazenados em uma lista tabu para impedir o retorno às soluções já visitadas. A memória de longo prazo contém uma história seletiva de soluções e seus atributos encontrados durante o processo de busca e é utilizada em estratégias de diversificação e intensificação.

A Busca Tabu se destaca ao avaliar todas as soluções da vizinhança e escolhe a melhor, desde que esta não esteja proibida ou contida na lista tabu, ou estando proibida, cumpra com o critério de aspiração. Este conceito, entretanto, carrega a possibilidade de se completar um ciclo, isto é, em cada ciclo a busca pode voltar a uma configuração já visitada. A fim de evitar este problema, a lista tabu é ajustada com informações das configurações para o processo de busca. Geralmente a lista tabu é usada para proibir aqueles movimentos da vizinhança que poderão cancelar o efeito de movimentos recen-

temente executados e conduz assim o processo de busca para trás a uma configuração anteriormente visitada.

Tipicamente, se o movimento correspondente da vizinhança leva a uma configuração melhor que a anterior o movimento é então aceito (critério de aspiração). A BT faz a varredura da vizinhança e aceita então a melhor configuração vizinha, até que nenhum dos vizinhos melhore o valor atual da função objetivo.

A BT explora a vizinhança de uma solução dada e seleciona a melhor solução encontrada nesta vizinhança mesmo que esta piore a solução corrente. Esta estratégia permite que a busca escape de um ótimo local e explore outra parcela do espaço de solução. Os mecanismos básicos de Busca Tabu são:

- a estrutura de memória,
- o critério de aspiração, e
- o critério de parada.

## 4.2 Princípios Básicos de Busca Tabu

Busca Tabu foi desenvolvido sobre um conjunto de princípios (funções) que de forma integrada, permitem resolver um problema de maneira inteligente. O princípio filosófico de Busca Tabu pode ser explicado com as próprias palavras de seu criador F. Glover (GLOVER, 1986) que transcreve-se aqui com pequenas modificações:

> *BT está baseada na premissa de que a resolução de um problema pode ser considerada inteligente se esse processo incorpora a memória adaptativa e a exploração sensível. O uso de memória adaptativa contrasta com as técnicas sem memória (como "Simulated Annealing" e o Algoritmo Genético) e com as técnicas de memória rígida (como as técnicas de Inteligência Artificial e de "Branch and Bound"). De igual maneira, a idéia de exploração sensível em BT está inspirada na suposição de que uma escolha ruim realizada por uma estratégia produz mais informação que uma boa escolha aleatória (numa crítica evidente, por exemplo, a SA que faz escolhas aleatórias). Assim, se a estratégia que guia um algoritmo que usa memória (como BT) faz uma escolha ruim (passa a uma configuração de baixa qualidade) então, pode-se aproveitar essa informação (escolha ruim) para evitar voltar a visitar essa configuração (ruim) e, ainda melhor, para modificar (melhorar) a própria estratégia que guia o processo de busca para ter capacidade de encontrar ou escolher configurações de melhor qualidade.*

Na Tabela 1 mostram-se os elementos principais (funções) de BT. Portanto, as pesquisas em BT consistem em usar estas funções de maneira integrada e eficiente para resolver cada problema específico, levando em conta que cada uma das funções podem ser implementadas de várias formas diferentes dependendo das características do problema e do nível de sofisticação da implementação do algoritmo BT. Outra linha de pesquisa muito ativa, consiste em criar outras funções para que sejam incorporadas como parte da estrutura da BT e/ou encontrar variantes de implementação das funções mostradas na Tabela 1. As características específicas de cada problema e a experiência do pesquisador devem determinar a forma e a qualidade do algoritmo BT implementado.

Apresenta-se brevemente a essência da lógica do algoritmo de Busca Tabu para resolver problemas genéricos do tipo:

$$\begin{aligned} Min \quad & f(x) \\ \text{s.a.} \quad & x \in X \end{aligned} \tag{4.1}$$

Busca Tabu resolve (4.1) iniciando com um processo similar a qualquer algoritmo heurístico de busca local. Na busca local, dada uma configuração $x$ (solução), define-

Tabela 1: Funções de Busca Tabu.

PRINCIPAIS CARACTERÍSTICAS DA BUSCA TABU

## Memória Adaptativa

Seletividade (incluido esquecimento estratégico)

Abstração e decomposição (usando memória explícita e por atributos)

Tempo:

Recência de eventos
Freqüência de eventos
Diferenciação entre curto e longo prazo

Qualidade e impacto:

Importância relativa quanto às escolhas alternativas
Impacto de mudanças de relações em estrutura ou restrições

Contexto:

Interdependência regional
Interdependência estrutural
Interdependência seqüencial

## Exploração Sensível

Imposição estratégica de proibições e induções
(*condições tabu e níveis de aspiração*)

Enfoque concentrado em boas regiões e em boas características das soluções
(*processo de intensificação*)

Caracterização e exploração de novas regiões promissoras
(*processo de diversificação*)

Padrões de busca não monótonos
(*oscilação estratégica*)

Integração e geração de novas soluções
(*path relinking*)

se uma vizinhança de $x$ como sendo o conjunto de todas as configurações $x' \epsilon N(x)$ que podem ser obtidas pela aplicação de um mecanismo de transição a partir de $x$. A estrutura de vizinhança define as condições para que $x'$ seja vizinho de $x$. Assim, por exemplo, no problema de planejamento de sistemas de transmissão se pode definir como vizinho de uma configuração $x$ todas aquelas configurações que podem ser obtidas a partir de $x$ pela adição de um circuito, a retirada de um circuito ou troca de dois circuitos (retirada de um circuito e adição de outro circuito). No algoritmo de busca local, a partir da configuração corrente, passa-se para a configuração vizinha que apresenta uma maior diminuição da função objetivo. Um procedimento repetitivo desta estratégia leva o algoritmo de busca local a parar no momento em que não existe nenhuma configuração vizinha que produza uma diminuição da função objetivo o que significa que foi encontrado um ótimo local. Entretanto, para uma BT não parar em ótimos locais, são empregadas técnicas de diversificação em regiões ainda não exploradas, intensificação em torno de configurações de boa qualidade, estratégias de *"path relinking"* entre duas configurações de boa qualidade, que pertencem a lista de elite armazenada no processo BT e a oscilação estratégica empregada quando o processo BT chega a um nível crítico. Cada uma destas técnicas será explicada em detalhe mais a frente.

## 4.3 Busca Tabu Dedicada ao Planejamento da Expansão de Transmissão

Nesta seção, é apresentado um exemplo de algoritmo BT dedicado ao PPET. Na Figura 1 mostra-se o módulo de funcionamento do algoritmo BT, em que o processo de otimização é iniciado com uma configuração inicial que pode ser obtida aleatoriamente, mas para sistemas de médio e grande porte é muito mais eficiente usar um algoritmo inicializador para encontrar uma boa configuração inicial.

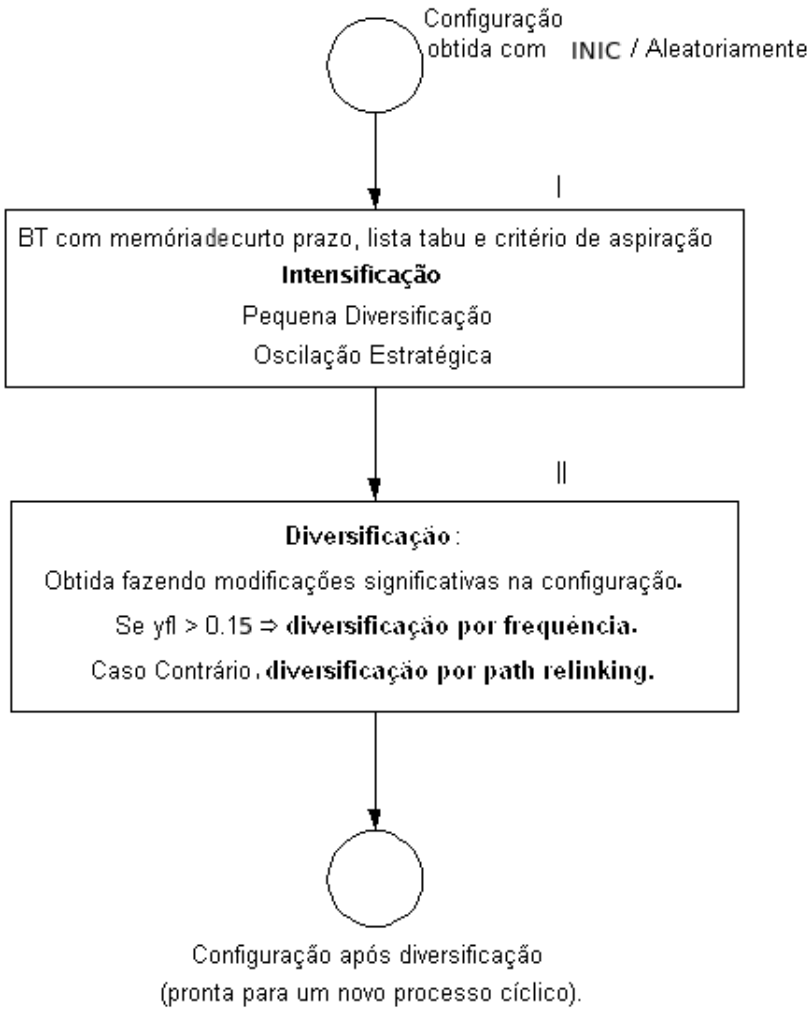



Figura 1: Mecanismo Busca Tabu (GALLEGO, 1997).

No módulo I, esta configuração é submetida a um processo de BT com memória de curto prazo que compreende de três partes claramente diferenciadas:

- Processo BT com memória de curto prazo, listas tabu e critério de aspiração;
- Intensificação;
- Diversificação.

No módulo II, é realizada uma estratégia de diversificação que pode ser implementada de várias formas e depois o processo pode retornar ao módulo I em um processo cíclico até que seja satisfeito um critério de parada. Portanto, a estratégia mostrada na Figura 1 corresponde a uma forma de integrar algumas das funções BT, para resolver o PPET.

Evidentemente, podem ser desenvolvidas outras estratégias BT para resolver um mesmo tipo de problema levando a versões diferentes do algoritmo BT. É importante analisar detalhadamente cada uma das partes da estratégia apresentada na Figura 1.

### 4.3.1 Determinação de uma Configuração Inicial

Foi verificado experimentalmente que os algoritmos combinatórios apresentam melhor desempenho quando os processos são iniciados com aceitáveis configurações iniciais geradas empregando um Algoritmo Heurístico Construtivo (AHC) em lugar de configurações obtidas aleatoriamente. Esta característica é mais importante ainda no caso do algoritmo BT, pois observa-se que boas configurações iniciais diminuem o esforço computacional do processo e freqüentemente encontram configurações de melhor qualidade. Para se obter uma melhor diversidade das configurações, é importante que as mesmas sejam obtidas através de diferentes heurísticas, pois assim as características peculiares à cada método estariam presentes no conjunto de configurações inciais utilizado. Neste trabalho, todas as configurações iniciais utilizadas nas simulações dos sistemas testes para os diferentes algoritmos de BT foram geradas previamente por um time assíncrono inicializador (DE-OLIVEIRA; DEALMEIDA; MONTICELLI, 1998) e (DEOLIVEIRA; DEALMEIDA; MONTICELLI, 1999).

### 4.3.2 Solução do Problema por BT

Pode-se implementar um algoritmo BT dividido em duas partes, que utilizam as estratégias de intensificação e diversificação apresentadas na Figura 1. Este algoritmo pode ser inicializado com uma configuração inicial ou com $K$ configurações iniciais (conjunto de configurações iniciais).

Na Figura 2 mostra-se a parte do algoritmo que inicia com uma única configuração de partida que é submetida a um processo BT/intensificação.

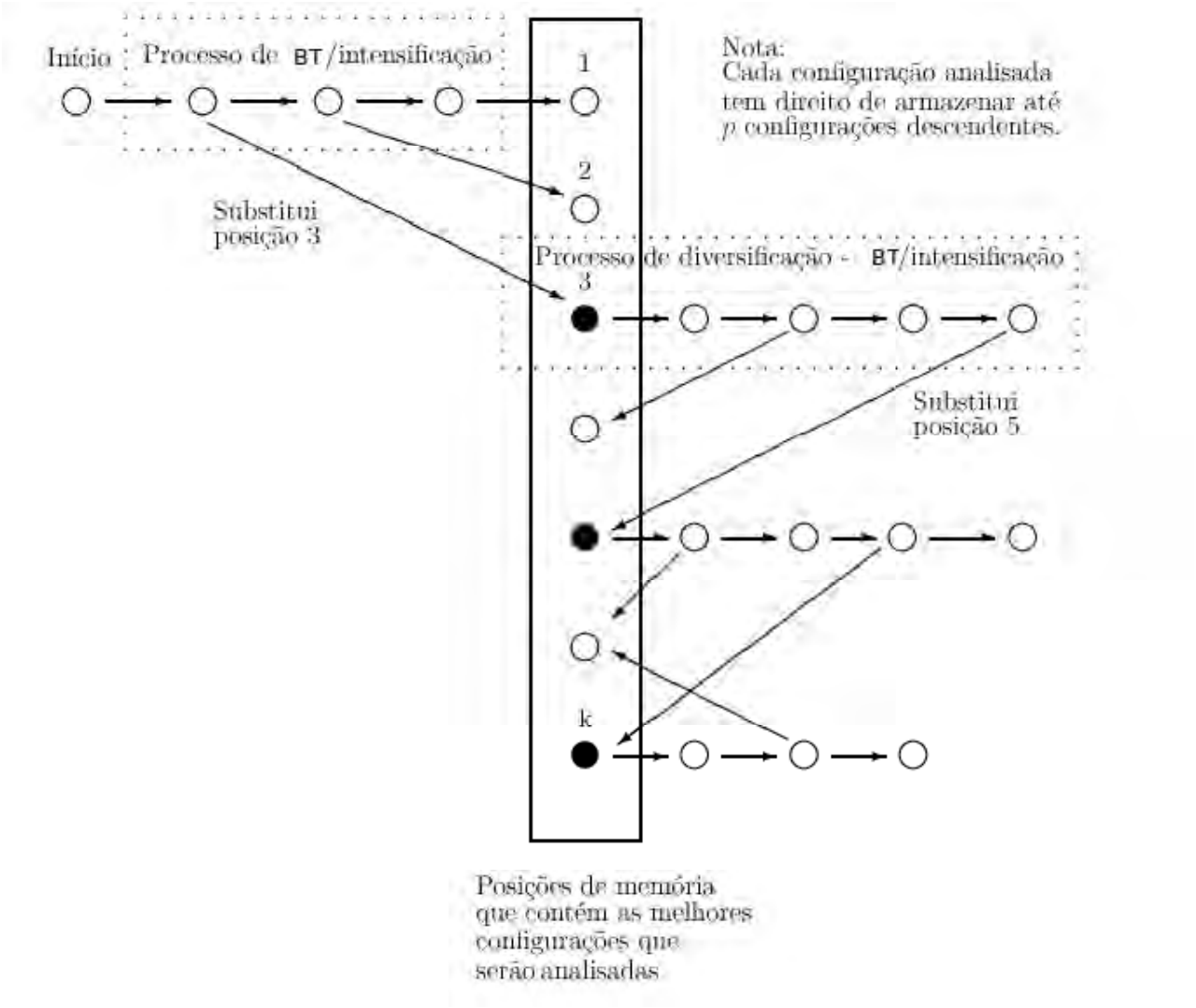



Figura 2: Algoritmo de Busca Tabu com uma única configuração inicial de partida (GALLEGO, 1997).

Durante este processo as $p$ melhores configurações são armazenadas. No passo seguinte, das configurações armazenadas é selecionada aquela de melhor função objetivo a qual é submetida a um processo de diversificação-BT/intensificação. Durante o processo de BT/intensificação, cada vez que se obtém uma configuração com pequeno corte de carga, é comparada sua função objetivo com as das armazenadas; se é melhor, substitui-se a de pior função objetivo. O número de configurações substituídas é limitado a $p$. Caso obtenha-se mais de $p$ configurações com melhor função objetivo, substitui-se algumas das $p$ anteriormente armazenadas. Selecionam-se das melhores configurações armazenadas, aquelas que formarão a *lista de elite*, a qual será empregado no processo de diversificação. Termina-se o processo quando todas as $K$ configurações armazenadas tiverem indicação

de que já foram bem avaliadas.

Na Figura 3 é apresentado mais detalhadamente o algoritmo para o caso de $K$ configurações iniciais. Este processo é dividido em dois níveis. No nível I, trabalha-se com um conjunto de $K$ configurações iniciais geradas conforme relatado anteriormente. Todas as configurações iniciais, passam pelo processo de BT/intensificação, cada configuração independentemente das outras. Já as configurações que fazem parte no nível II são as configurações iniciais após um processo de BT/intensificação.

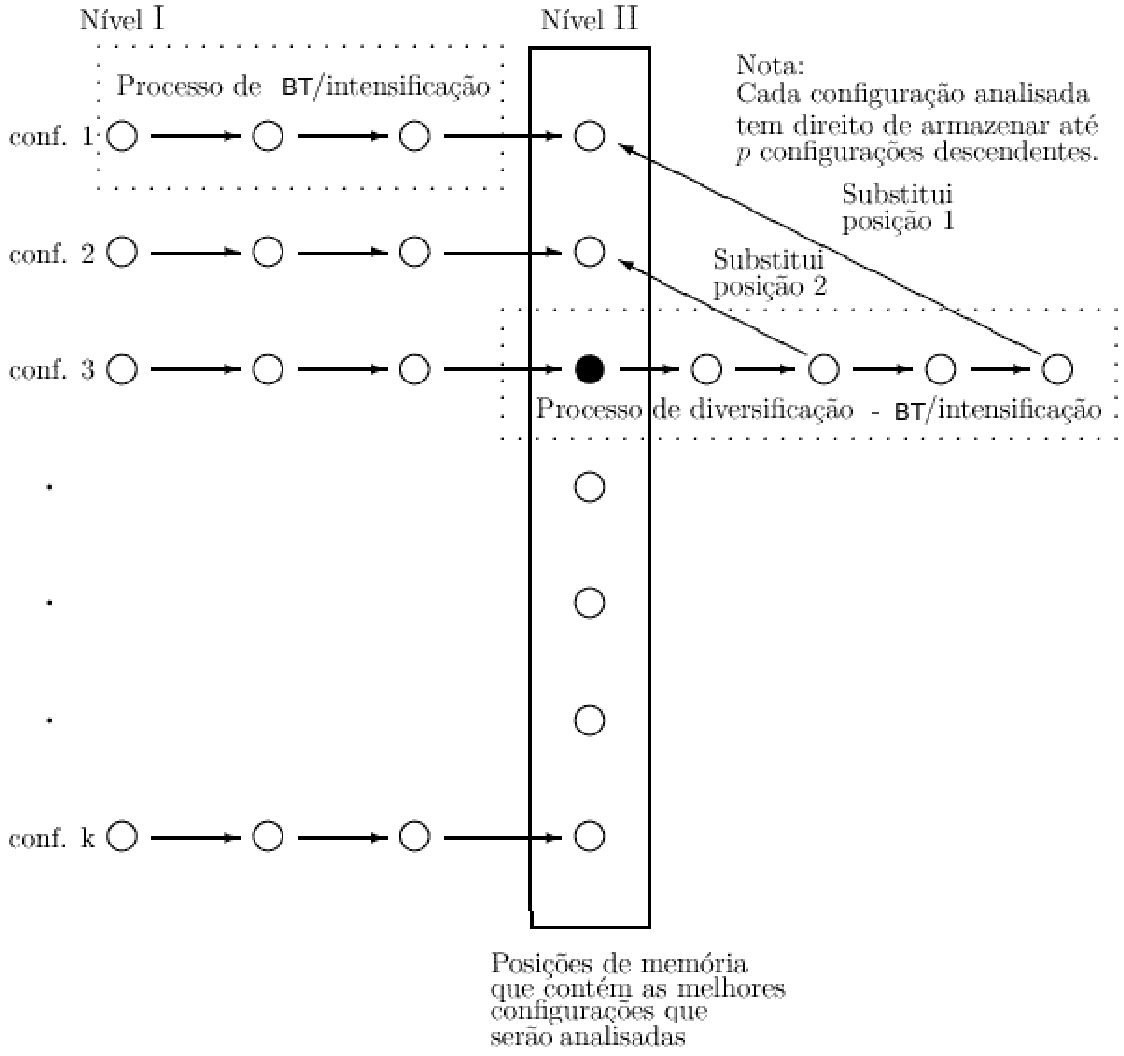



Figura 3: Algoritmo de Busca Tabu com $K$ configurações iniciais de partida (GALLEGO, 1997).

A partir do nível II foi realizado um processo de otimização integrado. Assim, antes de iniciar esta fase, deve-se construir a lista de elite formada por um subconjunto das melhores configurações. No nivel II cada uma das configurações são submetidas a um processo cíclico de diversificação e um processo BT/intensificação. O processo é iniciado

pela configuração com melhor função objetivo. Supondo que a terceira função objetivo tenha o melhor valor, assim esta configuração é submetida a um processo de diversificação seguido de um processo BT/intensificação. O processo de diversificação-BT/intensificação continua, em cada passo, escolhendo a configuração da lista que tem uma melhor função objetivo e que não tem a indicação de que foram analisadas ou se tem esgotado um número máximo de PPLs resolvidos.

### 4.3.3 Estrutura de Vizinhança Aplicada

Busca Tabu é um procedimento que guia um algoritmo de busca local para procurar o espaço de soluções além da simples otimalidade local. Isto se consegue através de uma busca agressiva, selecionando o melhor dos movimentos em cada passo.

Busca Tabu tem estratégias de busca especiais. Essa metaheurística se baseia em movimentos determinísticos em vez de aleatórios; então, neste caso, empregam-se diferentes tipos de memória e seleciona-se o melhor dos movimentos a cada iteração. A estrutura de vizinhança deverá restringir o número de linhas candidatas, já que uma estrutura com muitos vizinhos é onerosa do ponto de vista do tempo de computação gasto para avaliar cada uma das alternativas, que neste caso seria a resolução de um PPL para cada elemento da lista que é analisado. Assim, para melhorar este aspecto, usa-se então estratégias de listas de candidatas com a finalidade de restringir o número de soluções examinadas em uma dada iteração. As linhas candidatas deverão ser selecionadas de maneira cuidadosa; dessa forma regras eficientes para gerar as listas de candidatas são críticas neste processo.

Baseando-se nestas características, foi implementada uma estrutura de vizinhança que permita identificar os novos vizinhos; este processo de seleção dos vizinhos é feito através de listas elites de linhas candidatas. Existem então dois tipos de listas: para o corte de carga da configuração candidata maior ou menor que um limite estabelecido. Assim, se a configuração analisada tem um corte de carga maior que o limite estabelecido, a lista será formada por linhas a serem adicionadas/trocadas, caso contrário, quando o corte de carga é inferior ao limite, a lista será formada por linhas candidatas a serem retiradas/trocadas; este procedimento para a formação destas listas está baseado em métodos aproximados.

Nas listas de elite de linhas candidatas são empregadas tanto na etapa de BT intensificação como na etapa de BT diversificação, efetuando procedimentos de entrada/troca ou retirada/troca de linhas.

Nas listas elite de linhas, existem alguns vizinhos que são selecionados de maneira

aleatória tanto na etapa de intensificação como na de diversificação. Ou seja, na etapa de intensificação, se após aplicar os procedimentos de entrada/troca ou retirada/troca de linhas, com base nas listas de elite, não se obtém uma configuração com melhor função objetivo, são selecionados de maneira aleatória algumas linhas e se efetua um procedimento de troca. Na etapa de diversificação, além das diversificações feitas com base nas listas de elite, são efetuadas algumas trocas de linhas selecionadas de maneira aleatória.

### 4.3.4 Processo de Oscilação Estratégica

O processo de oscilação estratégica é mostrado na Figura 4, em que aparece evidente o mecanismo da oscilação estratégica. No processo de oscilação estratégica existem três processos que são indicados por memória de curto prazo, intensificação e diversificação e realizam as seguintes tarefas:

- A primeira parte corresponde ao processo BT com memória de curto prazo, listas tabu e critério de aspiração. Nesta parte do processo as transições são realizadas através de configurações infactíveis, a estratégia consiste em encontrar uma configuração factível de boa qualidade e, portanto, essa parte do processo termina quando essa configuração é encontrada. A forma de determinar os vizinhos candidatos segue uma estratégia diferente, como é apresentada posteriormente.

- A segunda parte corresponde a um conjunto de transições realizadas através de configurações factíveis. Assim, o segundo processo é uma intensificação em torno de uma boa configuração factível, portanto, o critério de vizinhança nesta parte do processo também deve ser diferente e deve permitir somente transições entre configurações factíveis, isto é, não são permitidas configurações com corte de carga.

- A terceira parte do processo corresponde a uma simples transição que deve produzir o retorno à região infactível. Assim, nesta parte do processo o critério de vizinhança também deve ser diferente, assim como a atualização das listas tabu. A idéia básica é sair da região factível para iniciar um novo retorno a região factível (novo processo com memória de curto prazo, mas as listas tabu devem proibir o retorno à região factível já visitada.)

  O processo cíclico das três partes do processo, que corresponde ao mecanismo de oscilação estratégica, deve ser implementado até satisfazer um critério de parada que pode ser um número máximo de resoluções de PPL e/ou uma ausência de melhoria

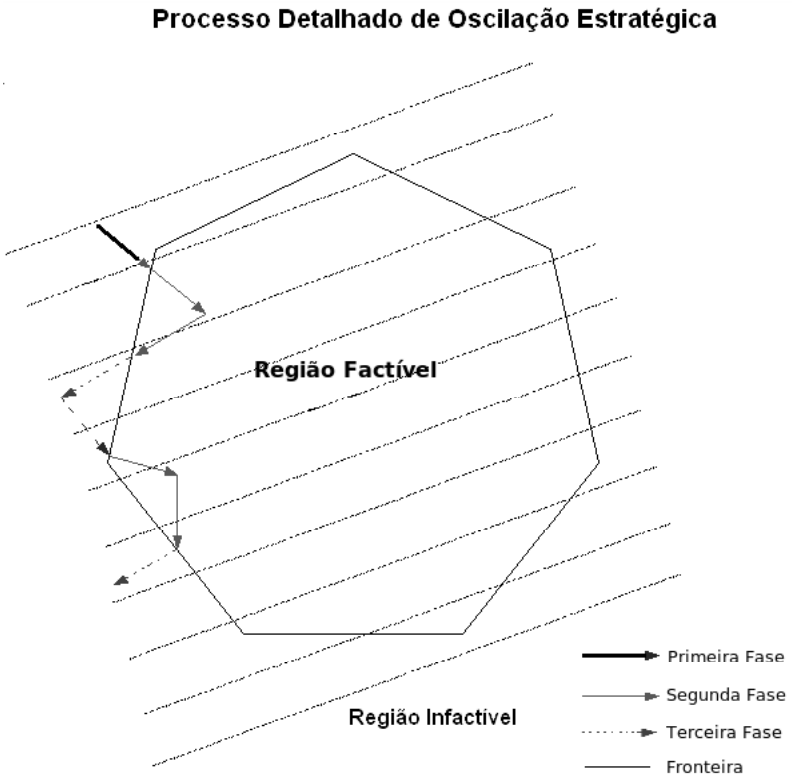



Figura 4: Estratégias empregadas no algoritmo.

na função objetivo em um número determinado de processos cíclicos. Nesse caso, deve-se implementar a diversificação proposta no módulo II, Figura 1.

### 4.3.5 Processo de Diversificação

Na Figura 1 o processo BT poderia terminar após o módulo I. Entretanto, processos mais sofisticados do algoritmo BT devem incorporar outras funções BT como a diversificação, o que é realizado no módulo II do algoritmo BT apresentado. Diversificação pode ser implementada de duas maneiras diferentes:

- realizando algumas modificações na configuração através de adições, retiradas e/ou trocas de circuitos ou penalizando atributos;
- fazendo *"path relinking"* ou usando valores por freqüência.

A primeira forma de diversificação consiste em realizar pequenas modificações na topologia da configuração através de adições, retiradas ou trocas de circuitos. Outra forma consiste em penalizar por algumas transições os atributos muito acionados durante

o processo integral, informação que está presente nas listas de memória baseadas em freqüência, para priorizar o uso de outros atributos pouco usados levando o processo a regiões não visitadas e potencialmente atrativas.

Uma outra forma de realizar diversificação é usando *"path relinking"*. *Path relinking* é uma função de BT, que permite gerar uma nova configuração usando como base duas ou três configurações de qualidade chamadas configurações de elite. Neste caso é realizado *path relinking* entre a configuração candidata a diversificação e uma das configurações de elite. A decisão de realizar uma das duas formas de diversificação pode ser realizada de forma aleatória ou de maneira alternada.

### 4.3.6 Processo de Intensificação

A *estratégia de intensificação* consiste em mudar os critérios de seleção das novas configurações candidatas, isto é, mudar $N^{**}(x)$ ou $(N^{*}(x))$ para incentivar a busca de novas configurações aproveitando a informação acumulada no processo BT. A intensificação pode ser implementada de várias formas diferentes, tais como:

- eliminando vizinho ou incorporando novo vizinho $x'$ para a configuração corrente $x$, tipicamente considerando configurações de elite ou configuração com esses atributos como sendo "vizinhos"de $x$;
- modificando a caracterização de configuração vizinha, isto é, redefinindo o conjunto $N(x)$;
- retornando às regiões atrativas para realizar uma busca mais intensa na vizinhança das mesmas;
- realizando intensificação por decomposição.

No próximo capítulo, será apresentado o algoritmo BT, voltado para o PPET, especificando no mesmo algumas das características citadas anteriormente.

# 5 ALGORITMO BUSCA TABU SERIAL

Neste Capítulo apresentam-se as técnicas usadas na construção do algoritmo de Busca Tabu (BT) para a resolução do problema de planejamento da expansão da transmissão de energia elétrica (GALLEGO, 1997). Inicialmente o algoritmo, a partir de uma configuração ou de um conjunto de configurações, gera diferentes configurações, usando métodos heurísticos construtivos previamente determinados nos parâmetros de calibração. O algoritmo também poder ser inicializado com todas as configurações zeradas e/ou com todos métodos heurísticos construtivos desabilitados, atuando a partir de configurações iniciais geradas por algoritmo inicializador.

A modelagem matemática utilizada foi o modelo DC; como já mencionado anteriormente, e esse modelo é o mais utilizado na área de planejamento de sistemas de transmissão, porque oferece quase sempre respostas factíveis para o problema.

Neste trabalho foram utilizadas funções de busca específicas e que foram implementadas neste algoritmo. Essas funções procuram encontrar soluções melhores que as já encontradas até o momento na literatura especializada, no entanto, elas especificam a direção da busca para que o algoritmo chegue a uma determinada região e procure uma solução cujo valor da função objetivo seja atrativo ou melhor que o valor da função atual.

Essas funções, conhecidas como diversificação e intensificação, foram usadas no algoritmo implementado a partir de uma versão para estações SUN (GALLEGO; DEOLIVEIRA, 1997. 6 p.), portado para ambiente Linux. Usou-se a intensificação em uma parte do algoritmo e em outra parte usou-se a diversificação. Além dessas técnicas de busca, o algoritmo BT usa memória, o que na literatura não se encontra em outros tipos de metaheurísticas, sendo uma vantagem oferecida pela técnica BT. Maiores detalhes dessa memória serão apresentados posteriomente na seqüência deste trabalho.

Para um melhor conhecimento da forma como a BT trabalha (algumas funções são usadas neste algoritmo, sendo algumas já mencionadas), apresenta-se agora dois tipos de

algoritmos BT: o algoritmo básico e o avançado, cada um com diferentes tipos de funções e memórias.

## 5.1 Algoritmo Busca Tabu Básico

O algoritmo básico apresenta as seguintes partes:

- Um mecanismo de geração de movimentos, isto é, dada a configuração corrente, deve-se definir e caraterizar a *vizinhança da configuração corrente* e, portanto, de acordo com a lógica BT, deve-se passar para a melhor configuração vizinha (ou para a menos pior).

- A formação da *lista tabu*, que armazena as configurações vizinhas que estão proibidas. Este mecanismo evita retornar para as configurações já visitadas (ciclagem). A escolha do tamanho da lista tabu é crítica no desenvolvimento BT. Normalmente a lista tabu é montada usando informação de memória de curto prazo (definida posteriormente).

- A definição do *critério de aspiração* que tenta contornar as proibições impostas pela lista tabu. Neste contexto, o critério de aspiração anula a proibição de uma configuração vizinha se for verificada que essa configuração candidata é altamente atrativa de acordo com os critérios de aspiração definidos.

## 5.2 Algoritmo Busca Tabu Avançado

Um algoritmo avançado deve levar em conta os seguintes aspectos:

- Montar uma estratégia adequada para gerar uma configuração inicial de boa qualidade ou, ainda melhor, um conjunto de configurações de boa qualidade e de composição diversificada. Assim, existiriam disponíveis configurações atrativas correspondentes a regiões distantes o que facilita a implementação eficiente das estratégias de intensificação e diversificação. Esta parte é implementada usando algoritmos heurísticos, que são muito rápidos em termos de esforço computacional.

- Armazenamento das chamadas configurações de elite que podem ser posteriormente usadas para implementar a intensificação e a diversificação, e também na imple-

mentação do chamado *"path relinking"* que é usado para gerar novas configurações de qualidade, misturando adequadamente atributos das configurações de elite.

- Implementação de *estratégias de intensificação* e *diversificação*. Estas estratégias são basicamente implementadas usando as memórias baseadas em freqüência. A intensificação tenta explorar, mais intensamente, regiões onde foram encontradas excelentes configurações, e a diversificação tenta levar o processo de otimização a regiões distantes ou pouco exploradas na tentativa de encontrar novas configurações de excelente qualidade.

- Montar uma estratégia destinada a reduzir o número de vizinhos candidatos. Em BT, dada uma configuração, existe um número de configurações vizinhas que devem ser analisadas e a BT passa à melhor configuração vizinha. Se o número de vizinhos é elevado, a BT pode precisar de um elevado esforço computacional para realizar um movimento simples. Existem várias propostas para reduzir o número de vizinhos que devem ser analisadas na BT. As principais propostas são chamadas critério de aspiração *"plus"* e a formação de uma *lista de vizinhos considerados de elite*. Para problemas específicos, pode-se ainda montar uma lista reduzida, aproveitando as características específicas do problema.

Um algoritmo BT é diferente de um algoritmo de busca local em dois aspectos fundamentais:

- A partir da configuração corrente, passa-se à melhor configuração vizinha ou à menos pior, que implica que é permitida uma degradação da qualidade da função objetivo.

- O conjunto de vizinhos de $x$ não se carateriza de maneira estática. Assim, BT define uma nova estrutura de vizinhança, $N^x(x)$ que varia dinamicamente em estrutura e tamanho durante todo o processo de otimização. Esta estratégia permite a BT realizar uma busca eficiente e inteligente.

## 5.3 Memória de Curto Prazo

Um algoritmo BT pode ser implementado simplesmente usando a estratégia de curto prazo e muitas das pesquisas iniciais foram implementadas desta forma. A memória de curto prazo é basicamente a informação de atributos de configurações que foram modificados no passado recente. Esta informação é conhecida como memória baseada em

recência *("recency")*. A idéia desta estratégia é considerar todos os atributos selecionados no passado recente como sendo proibidos, portanto, todas as configurações candidatas que possuem algum dos atributos proibidos (tabu-ativos) são excluídos na formação do conjunto $N^x(x) \subseteq N(x)$ de configurações vizinhas para avaliação.

A idéia básica da memória baseada em recência é evitar revisitar configurações já visitadas e aquelas configurações que compartilham os atributos tabu-ativos. Esta última parte pode representar um problema, pois podem ser eliminadas $N^x(x)$ configurações muito atrativas. O critério de aspiração contorna em parte esta limitação.

## 5.4 Memória de Longo Prazo

A incorporação de uma estratégia de longo prazo fornece ao algoritmo BT uma sofisticação adicional e um desempenho melhor na maioria das aplicações. Isto significa que além da memória de curto prazo (estratégia de todo algoritmo BT básico) é incorporada uma estratégia adicional chamada estratégia de longo prazo.

Existem três aspectos fundamentais relacionados com a memória de longo prazo: (1) a memória baseada em freqüência, (2) a estratégia de intensificação, e (3) a estratégia de diversificação.

A memória baseada em freqüência consiste basicamente em armazenar a informação do número de vezes em que o atributo foi escolhido para gerar ou participar das configurações durante o processo BT. Existem dois tipos de memória baseadas em freqüência: a *freqüência de transição* que armazena o número de vezes em que um atributo é retirado ou adicionado para formar novas configurações, e a *freqüência de residência* ou *permanência* que armazena a informação do número de vezes em que o atributo permanece nas novas configurações ou em todas as configurações geradas durante o processo BT.

Estratégia de intensificação, consiste em mudar os critérios de seleção das novas configurações candidatas, isto é, mudar $N^{xx}(x)$ (ou $N^x(x)$) para incentivar a formação de novas configurações aproveitando a informação acumulada no processo BT. $N^{xx}(x)$ é um subconjunto reduzido de $N(x)$ obtido usando algum critério de redução de configurações candidatas.

A estratégia de diversificação foi projetada para levar o processo de busca para regiões novas e atrativas. Esta estratégia é implementada mudando a definição de vizinhança ou de configurações candidatas $N(x)$, incorporando "vizinhos" constituídos por atributos que

foram pouco usados. Nesta parte podem ser usadas as estruturas de memória usadas em intensificação ou em memória de curto prazo mas com critérios de duração modificados.

Na Figura 5 tem-se um exemplo clássico de memória a longo prazo baseado em recência, usando o processo de busca diversificação.

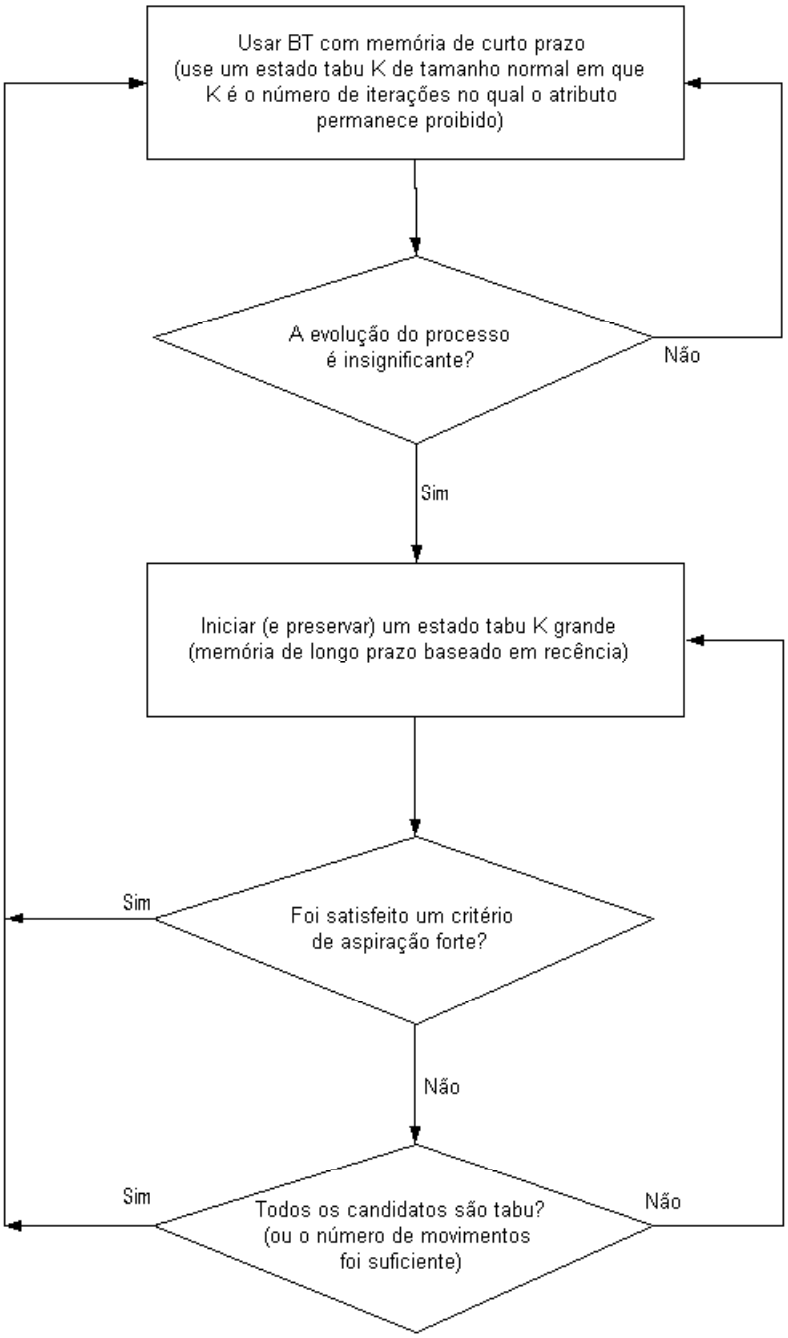



Figura 5: Diversificação usando memória de longo prazo baseado em recência (GALLEGO, 1997).

## 5.5 Algoritmo TSNOR

Na Figura 6, é apresentado o fluxograma do algoritmo de BT serial (TSNOR) que servirá como referência para a análise de desempenho dos algoritmos BT paralelos implementados. Neste algoritmo encontram todas as funções utilizadas de BT.

O algoritmo de BT é dividido em duas partes. Na primeira parte estão a memória de curto prazo, as listas tabu e o critério de aspiração, e como estratégia de busca utiliza-se a intensificação. Na segunda parte, estão a memória a longo prazo e a diversificação. A estratégia de busca diversificação pode ser realizada de duas maneiras: por *"path relinking"* ou freqüência; a decisão de realizar uma das duas formas de diversificação é decidida de forma aleatória. Assim a diversificação é realizada com *"path relinking"* com uma determinada probabilidade (tipicamente entre 0,1 e 0,2) e, caso contrário, é realizado o tipo de diversificação por freqüência.

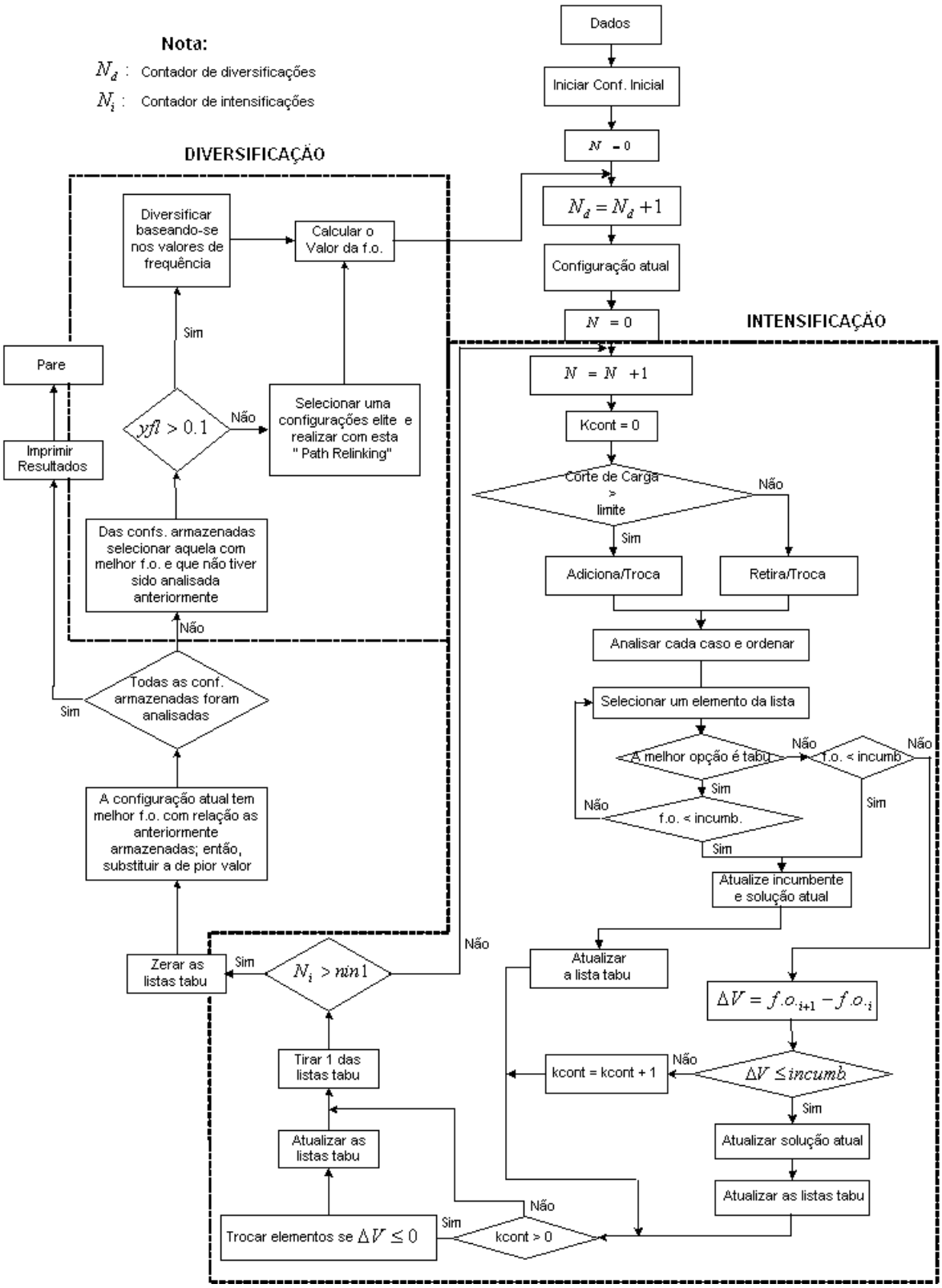



Figura 6: Diagrama de fluxo do algoritmo de BT serial (GALLEGO, 1997).

# 6 COMPUTAÇÃO PARALELA E MPI

## 6.1 Introdução

Nos sistemas computacionais tradicionais cada instrução é executada seqüencialmente, uma após a outra, pela única Unidade Central de Processamento **(CPU)** que compõe a máquina. No entanto, desde o desenvolvimento dos primeiros computadores, sempre busca-se uma forma alternativa de executar mais instruções simultaneamente. O objetivo sempre foi aumentar o poder computacional e a velocidade de processamento dos computadores para que aplicações complexas pudessem ser resolvidas com o auxílio da computação. Nesse contexto, surgiu a computação paralela que é vista como uma evolução da computação seqüencial. Assim como no mundo real, onde eventos complexos e intimamente relacionados ocorrem a todo momento simultaneamente, a computação paralela busca simular esse paralelismo de ações.

Basicamente, pode-se dizer que a computação paralela consiste no uso simultâneo de múltiplos recursos computacionais, como por exemplo processadores. Normalmente, os problemas podem ser divididos em partes menores e independentes e resolvidos separadamente, em paralelo, através de múltiplos processos distribuídos entre os processadores disponíveis.

Um dos motivos que impulsionaram o desenvolvimento da computação paralela foi a possibilidade de resolver aplicações complexas até então intratáveis com o auxílio da computação tradicional. Por volta dos anos 80, acredita-se que isso seria possível com o advento de processadores cada vez mais rápidos e eficientes no entanto, logo notou-se que isso não bastaria e começou-se a investir em um novo modelo de computação de processamento paralelo. Essa nova abordagem de desenvolvimento tornou-se uma opção para aqueles que desejam obter alto desempenho em seus sistemas computacionais. Hoje em dia, não só aplicações cientifícas, mas também comerciais (exploração de petróleo,

banco de dados, dentre outras) estão incentivando o desenvolvimento da computação paralela.

Neste capítulo, descreve-se os principais modelos, paradigmas e ambientes de programação paralela e discute-se as características do padrão MPI. Além disso, apresenta-se às principais rotinas definidas por esse padrão e mostram-se alguns fatores que influenciam no desempenho de programas MPI.

## 6.2 Modelos de Programação Paralela

Os modelos de programação paralela existem como uma camada de abstração sobre a arquitetura do hardware e da memória do computador (BARNEY, 2006). No entanto, esses modelos não são específicos de uma determinada arquitetura nem de um tipo de memória. Na literatura encontram-se vários modelos de programação paralela, porém, os mais comuns são (BARNEY, 2006) e (MOURA; SILVA; BUYYA, 1999):

- memória compartilhada;
- threads;
- paralelismo de dados;
- troca de mensagem;
- híbrido.

Geralmente, a escolha do modelo a ser utilizado depende das habilidades e preferências do programador, do tipo de hardware disponível e das características da aplicação. Não pode-se dizer que existe um modelo melhor que o outro, o que existe são melhores implementações para cada modelo (BARNEY, 2006). Nas próximas seções descreve-se, sucitamente, cada um dos modelos mencionados anteriormente.

### 6.2.1 Memória Compartilhada

Esse modelo de programação é caracterizado pela existência de um espaço de endereçamento comum (memória), compartilhado por todas as tarefas que estão sendo executadas no sistema computacional. No modelo de memória compartilhada, as tarefas podem ler e escrever na memória sem nenhuma restrição. Por isso, visando manter a

consistência e a coerência das informações armazenadas, precisa-se utilizar mecanismos de sincronização, como por exemplo semáforos, para controlar o acesso à memória.

### 6.2.2 *Threads*

No modelo baseado em *"threads"*, os processos podem ter vários caminhos possíveis de execução concorrente. Cada caminho de execução recebe o nome de *"thread"*. Apesar de uma thread possuir suas próprias variáveis locais, ela também compartilha recursos do sistema com outras *"threads"*.

Considerando um programa com várias subrotinas, nesse modelo pode-se associar uma *"thread"* a cada uma delas. Com isso, todas as subrotinas podem ser executadas ao mesmo tempo, de forma concorrente. Normalmente, o modelo de programação baseado em *"threads"* é utilizado em arquiteturas de memória compartilhada. Para que as *"thread"* se comuniquem, elas utilizam a memória global do sistema, lendo e escrevendo sobre variáveis compartilhadas. Assim como no modelo de memória compartilhada, também é preciso utilizar mecanismos de sincronização para evitar que duas ou mais *"thread"* escrevam, simultaneamente, sobre uma mesma posição de memória.

### 6.2.3 Paralelismo de Dados

A principal característica desse modelo é permitir que as tarefas sejam executadas em paralelo sobre elementos diferentes de uma estrutura de dados regular, como por exemplo vetor ou matriz. Suponha que deseja-se somar cinco unidades a cada elemento de um vetor **A**, que contém trinta posições (A[30]) e tem-se três processos disponíveis para realizar essa operação. Com isso, podemos dividir igualmente o vetor **A** entre os processos e determinar quais posições cada um irá somar cinco unidades. Assim, cada processo executa o mesmo programa sobre partes distintas do vetor **A**. Essa característica recebe o nome de SIMD *("Single Instruction, Multiple Data")*. Normalmente, aplicações científicas manipulam grandes estruturas de dados regulares e, por isso, esse modelo é bastante adequado e eficiente para ser utilizado nessa classe de aplicação.

### 6.2.4 Troca de Mensagem

As bibliotecas de troca de mensagem permitem escrever programas paralelos em ambientes de memória distribuída. Nesse modelo de programação, utilizam-se primitivas de comunicação explícitas, do tipo *send* e *receive*, para transmitir dados através de uma rede

de interconexão. Tipicamente, esse modelo de programação é escolhido quando pretende-se desenvolver programas paralelos para *"clusters"* de PCs.

Além de oferecer primitivas de comunicação, essas bibliotecas ainda possuem rotinas para inicializar e configurar o ambiente de execução. Nesse ambiente, um conjunto de tarefas é executado simultaneamente por processos distintos, distribuídos entre as máquinas disponíveis. Cada processo utiliza a memória local da máquina em que está rodando. As duas principais bibliotecas de troca de mensagem são: **PVM** *("Parallel Virtual Machine")* e **MPI** *("Message Passing Interface")* (MPIFORUM, 1995).

### 6.2.5 Híbrido

Algumas vezes se quer combinar características de modelos diferentes para desenvolver programas paralelos. Quando isso é desejável, é preciso utilizar um modelo de programação híbrido, que agrupe características de interesse.

Um modelo híbrido, muito utilizado no desenvolvimento de programas paralelos, combina o modelo de troca de mensagem com os modelos de *"threads"* ou de memória compartilhada. Um outro modelo híbrido, bastante comum, combina o paralelismo de dados com o de troca de mensagem. Alguns modelos, como o *"High Performance Fortran*" - HPF (HPFF, High performance FORTRAN: conceitos e definições: 2006), utilizam o paralelismo de dados sobre arquiteturas de memória distribuída. No entanto, esse modelo utiliza troca de mensagem para transmitir os dados entre as tarefas, de forma transparente ao programador.

## 6.3 Paradigmas de Programação Paralela

Existem vários paradigmas de programação paralela que podem ser escolhidos pelo programador para estruturar ou organizar o desenvolvimento de programas. A escolha de um ou de outro paradigma depende das características da aplicação, dos recursos computacionais disponíveis para quem vai desenvolver o programa e do tipo de paralelismo encontrado no problema. Nas próximas seções, será explicado, resumidamente, alguns dos paradigmas mais comumente utilizados na implementação de programas paralelos (MOURA; SILVA; BUYYA, 1999).

### 6.3.1 Modelo Mestre/Escravo

Esse paradigma é caracterizado pela existência de duas entidades fundamentais: uma mestre e outra escrava. A idéia desse paradigma é atribuir ao mestre a característica de coordenador da execução das atividades e aos escravos a função de executar as tarefas e as ordens do processo mestre.

Normalmente, o processo mestre divide uma tarefa em subtarefas e as distribui entre os escravos existentes. Em seguida, cada um dos escravos deve resolver suas tarefas e retornar os resultados ao mestre. Depois de receber os dados enviados por todos os escravos, o processo mestre se encarrega de produzir o resultado final da computação. Vale ressaltar que, normalmente, todas as comunicações existentes na aplicação acontecem entre entidades distintas (mestre e escravo) e não entre os próprios escravos ou mestres.

Os processos escravos podem estar espalhados em várias máquinas e serem submetidos a diferentes cargas de trabalho. Por isso, visando distribuir melhor os trabalhos entre os escravos, esse paradigma pode utilizar algum tipo de balanceamento de carga. O balanceamento de carga, estático ou dinâmico, pode infuenciar o desempenho das aplicações.

No balanceamento estático, a divisão e a distribuição das tarefas são feitas no início do programa e conta com a participação do mestre no processo de computação. Já o balanceamento dinâmico pode ser aplicado quando existe a presença de nós de processamento saturado, quando não se conhece o número de tarefas criadas no início da execução do programa a serem distribuídas ou no caso em que o número de tarefas a serem executadas ultrapassa a quantidade de processadores disponíveis no sistema.

As vantagens obtidas no uso do paradigma mestre/escravo são: o alto grau de escalabilidade alcançado e o *speedup* normalmente obtido. Por outro lado, deve-se tomar alguns cuidados para que o controle centralizado atribuído ao mestre não se torne um gargalo para o sistema. Em situações com grande quantidade de processadores disponíveis, a existência de um único mestre pode prejudicar o desempenho do sistema. Para que isso não ocorra, pode-se aumentar o número de processos mestres e fazer com que cada um deles cuide de um grupo distinto de processos escravos.

### 6.3.2 Modelo *Single Program Multiple Data* (SPMD)

Paralelismo de dados (*"Single Program, Multiple Data"*) - SPMD: uma mesma tarefa é alocada aos diversos processadores sendo que cada nó trabalha com uma porção de dados

diferentes do outro nó. Desta forma, divide-se o espaço de busca da solução conforme a disponibilidade dos processadores. Cada processador trabalha com o mesmo código do programa, mas as tarefas alocadas a cada processador podem ser diferentes. O valor do *rank* é predeterminado, e cada processador roda a instância associada ao seu *rank*. Esta abordagem é aplicada a alguns problemas clássicos:

- Resolução de sistemas de equações;
- Multiplicação de matrizes;
- Integração numérica;
- Mineração de dados.

Neste trabalho é utilizado este modelo de paradigma de programação em todos os algoritmos paralelos desenvolvidos.

### 6.3.3 Modelos Híbridos

Um modelo híbrido é construído e utilizado quando uma determinada aplicação permite explorar, com eficiência, mais de um tipo de paralelismo presente em suas características. Normalmente, esses paradigmas são aplicados na resolução de problemas de grande porte. Assim, conceitos e estratégias de diferentes paradigmas são empregados para estruturar e propor uma boa solução computacional.

## 6.4 Ambiente de Programação Paralela

Para que se possa utilizar e explorar todos os recursos disponíveis em um sistema de computação paralelo, deve-se conhecer as linguagens e os ambientes de programação disponíveis, além dos modelos e paradigmas de programação existentes. Esse conhecimento é importante para se escrever programas paralelos eficientes. A classificação dos ambientes estão baseadas nos paradigmas de programação que podem ser utilizados em cada um deles. Na seqüência, apresentam-se os dois ambientes de programação paralela mais conhecidos (PRAMANICK, 1999).

### 6.4.1 Ambientes de Memória Compartilhada

Os ambientes de memória compartilhada são caracterizados pela presença de vários processadores, compartilhando o acesso a uma única memória. Os processadores podem funcionar de forma independente, mas qualquer mudança no conteúdo das variáveis armazenadas na memória é visível a todos os outros processadores.

Baseado no tempo de acesso à memória, gasto por cada processador pertencente ao sistema, pode-se dividir as máquinas de memória compartilhada em duas classes: UMA( *"Uniform Memory Access"*) e NUMA ( *"Non Uniform Memory Access"*).

Nas máquinas pertencentes à arquitetura UMA, o tempo gasto no acesso à uma mesma posição de memória é igual para qualquer processador. Infelizmente, com o aumento natural do número de processadores que compõem essas máquinas, o barramento de acesso à memória pode ficar saturado e se tornar um gargalo para o sistema.

A arquitetura NUMA surgiu para solucionar a saturação de barramento mencionada anteriormente. Cada processador possui um módulo de memória associado, utilizado somente por tarefas locais. O conjunto de todos esses módulos de memória formam a memória global do sistema. Ao contrário das máquinas da arquitetura UMA, nessa classe de memória compartilhada os processadores não possuem o mesmo tempo de acesso à memória. A comunicação entre os processadores acontece através da leitura e escrita de dados na memória compartilhada pelo sistema.

O ambiente de memória compartilhada oferece algumas vantagens e desvantagens sobre os demais ambientes de programação paralela (BARNEY, 2006). Como principais vantagens, pode-se citar:

- A existência de um espaço de endereçamento global torna a programação nesse ambiente bastante amigável para o programador;
- Como a memória está próxima às CPUs, o compartilhamento dos dados entre as tarefas é rápido e uniforme.

Por outro lado, cita-se como desvantagens:

- A baixa escalabilidade do número de processadores, uma vez que o aumento demasiado de CPUs pode congestionar o barramento de acesso à memória;
- A dificuldade e necessidade em manter a coerência de *cache*;

- A sincronização entre os acessos à memória global é de responsabilidade do programador;
- O preço para projetar e produzir máquinas de memória compartilhada é ainda muito alto.

### 6.4.2 Ambiente de Memória Distribuída

Neste trabalho, usa-se a memória distribuída; nesse ambiente cada processador possui sua própria memória local e não existe uma memória compartilhada pelo sistema. Com isso, os processadores podem trabalhar independentemente, acessando somente sua memória local sem afetar os dados utilizados pelos demais processadores.

Eventualmente, um processador precisa acessar dados armazenados na memória local de outros processadores. Quando isso for necessário, cabe ao programador definir, explicitamente, como e quando o dado será acessado. Para isso, barreiras de sincronização entre as tarefas que estão sendo executadas em cada processador devem ser definidas, afim de coordenar as atividades.

Esses ambientes têm impulsionado a utilização dos paradigmas de troca de mensagem, tais como o PVM e o MPI. Esses paradigmas utilizam, explicitamente, primitivas de comunicação, como *send* e *receive*, para realizar a transferência de dados ou de mensagens entre os processadores distribuídos pelo sistema.

Esse ambiente também possui algumas vantagens e desvantagens em relação aos demais (BARNEY, 2006). Como vantagens cita-se:

- A quantidade de memória do sistema aumenta com a adição de novas *CPUs*, podendo melhorar o desempenho das aplicações. Para inserir um novo processador ao sistema computacional também é necessário adicionar uma memória local;
- Cada processador pode acessar rapidamente sua memória local, sem nenhuma interferência dos demais processadores;
- Ausência da necessidade de manter a coerência de *cache*.

As desvantagens encontradas nesse ambiente são:

- O tempo de acesso à memória não é uniforme (NUMA);

- Muitos detalhes associados à comunicação existente entre os processadores ficam sob a responsabilidade do programador;
- A dificuldade em mapear estruturas de dados já existentes em ambientes de memória global.

Na próxima seção são apresentadas algumas características da interface de troca de mensagem MPI, utilizada com maior freqüência nos ambientes de memória distribuída (e usada neste trabalho), além de suas principais primitivas de comunicação.

## 6.5 Programação com Troca de Mensagem (MPI)

Esse paradigma de programação paralela, ao contrário do que muitas pessoas pensam, não está restrito às aplicações acadêmicas e científicas. O interesse comercial pela programação com troca de mensagens é antigo. Os primeiros computadores paralelos para fins comerciais utilizavam o paradigma de troca de mensagens no desenvolvimento das aplicações. Assim, visando interesses de cada fabricante, muitas bibliotecas de troca de mensagens foram implementadas para um hardware específico. As principais diferenças encontradas entre as implementações estavam relacionadas à sintaxe das primitivas de cada biblioteca.

Por esses motivos, houve a necessidade de criar uma biblioteca padrão que pudesse ser utilizada sobre o hardware de diferentes fabricantes. Com esse intuito surgiu inicialmente o PVM e em seguida o MPI, definido como um padrão para os programadores e usuários de bibliotecas de troca de mensagens. O MPI é capaz de prover um ambiente prático, portável, eficiente e flexível para o desenvolvimento de aplicações paralelas de alta performance (AL-TAWIL; MORITZ, 2001) e (BARNEY, 2002). Por esses e outros motivos, ele tem se tornado um padrão emergente para a implementação de programas paralelos sobre os ambientes de memória distribuída (NUPAIROJ; NI, 1994).

A definição do padrão MPI contou com a participação de mais de 40 organizações, dentre elas: fabricantes de hardware, programadores e pesquisadores. O principal objetivo desse padrão é oferecer uma biblioteca de programação eficiente e amplamente portável, sem prejudicar o desempenho do programa (AL-TAWIL; MORITZ, 2001). Além disso, esse padrão é visto como uma camada de comunicação flexível, que dispõe de vários mecanismos para a transferência de mensagems, ponto-a-ponto ou coletivas (NUPAIROJ; NI, 1994) e (AL-TAWIL; MORITZ, 2001).

Desde a definição do padrão, várias implementações surgiram e se tornaram publicamente disponíveis, tais como: CHIMP (ALASDAIRA et al., 1994), LAM (BURNS; DAOUD; VAIGL, 1994), MPICH (GROPP; LUSK; SKJELLUM, ). Apesar dessa variedade de implementações, todas apresentam um desempenho semelhante quando comparadas à uma mesma plataforma e configurações (NUPAIROJ; NI, 1994). Neste trabalho, utilizou-se a implementação LAM nos testes experimentais.

Nessa abordagem, os programas são constituídos por um conjunto de processos, que podem estar distribuídos em diferentes máquinas, executando trechos de códigos independentes ou cooperando na resolução de um problema. Para que isso seja possível, os processos trocam informações (dados) através de primitivas de comunicações disponibilizadas pelo MPI.

Normalmente, esse tipo de paradigma de programação é utilizado sobre redes de estações de trabalhos (NOWs) ou *"clusters"*. Nessas plataformas, as trocas de mensagens entre cada um dos processadores/nós são utilizadas tanto para a transferência de dados, quanto para sincronizar tarefas.

No MPI, as duas operações básicas para trocas de mensagens são: *send* e *receive*. Para escrever qualquer programa paralelo, ainda é necessário ter uma função capaz de especificar a quantidade de processos que estão participando da computação e uma outra capaz de retornar um identificador associado a cada processo. Essas funções são chamadas de *MPI_COMM_SIZE* e *MPI_COMM_RANK*, respectivamente. As demais operações surgiram para facilitar a implementação de algumas comunicações de comportamentos distintos, que apresentam características específicas das aplicações (um *broadcast*, por exemplo).

As primitivas de comunicação utilizam o processamento disponível da CPU para transmitir os dados entre os diferentes nós de um *"clusters"* e, conseqüentemente, acabam "roubando" o processamento que poderia estar sendo utilizado para realizar as computações locais. No entanto, as plataformas recentes oferecem um suporte adicional, através do *"hardware"*, para enviar e receber mensagens. O *"hardware"* da interface de rede suporta o DMA (*"Direct Memory Access"*) e a transferência de mensagem assíncrona. Essa interface de rede possibilita a transferência das mensagens de um buffer de memória para o buffer de destino, sem qualquer intervenção da CPU. Com isso, os processos não precisam interromper a sua computação local para enviar ou receber uma mensagem.

Dentre as razões para usar-se o MPI nas implementações de programas paralelos, indicam-se as seguintes, citadas por (BARNEY, 2002) e (MPIFORUM, 1995):

- padronização: o MPI foi definido como um padrão para implementação de bibliotecas de troca de mensagem;
- portabilidade: não existe a necessidade de modificar o código fonte de um programa MPI para portar uma aplicação a outras plataformas;
- desempenho: fabricantes de hardware podem explorar as características do próprio equipamento para melhorar o desempenho de programas MPI;
- funcionalidade: existem mais de 115 rotinas definidas no padrão;
- disponibilidade: muitas implementações do padrão podem ser encontradas, seja elas de domínio público ou particular.

Originalmente, o MPI foi projetado e desenvolvido para sistemas de memória distribuída. Hoje em dia, ele também está sendo utilizado para implementar alguns modelos de memória compartilhada, como o paralelismo de dados, sobre arquiteturas de memória distribuída (BARNEY, 2002).

No padrão MPI, uma aplicação é constituída por um ou mais processos que se comunicam, adicionando-se funções para o envio e recebimento de mensagens entre os processos. Inicialmente, na maioria das implementações, um conjunto fixo de processos é criado. Porém, esses processos podem executar diferentes programas. Por isso, o padrão MPI é algumas vezes referido como SPMD *("Single Program, Multiple Data")*.

Elementos importantes em implementações paralelas são a comunicação de dados entre processos paralelos e o balanceamento da carga. Dado o fato do número de processos no MPI ser normalmente fixo, neste texto é enfocado o mecanismo usado para comunicação de dados entre processos. Os processos podem usar mecanismos de comunicação ponto a ponto (operações para enviar mensagens de um determinado processo a outro). Um grupo de processos pode invocar operações coletivas *("collective")* de comunicação para executar operações globais. O MPI é capaz de suportar comunicação assíncrona e programação modular, através de mecanismos de comunicadores *("communicator")* que permitem ao usuário MPI definir módulos que encapsulem estruturas de comunicação interna.

Os algoritmos que criam um processo para cada processador podem ser implementados, diretamente, utilizando-se comunicação ponto a ponto ou coletivas. Os algoritmos que implementam a criação de tarefas dinâmicas ou que garantem a execução concorrente de muitas tarefas, num único processador, precisam de um refinamento nas implementações com o MPI.

### 6.5.1 MPI - Básico

Embora o MPI seja um sistema complexo, um amplo conjunto de problemas pode ser resolvido usando-se apenas seis funções, que servem basicamente para: iniciar, terminar, executar, identificar processos, enviar e receber mensagens. Ver Tabela 2.

Tabela 2: Funções Básicas - MPI

| | |
|---|---|
| MPI_INIT | Inicia uma execução MPI |
| MPI_FINALIZE | Finaliza a execução |
| MPI_COMM_SIZE | Determina o número de processos |
| MPI_COMM_RANK | Determina a identificação dos processos |
| MPI_SEND | Envia mensagens |
| MPI_RECV | Recebe mensagens |

Todos os procedimentos na Tabela 2, exceto os dois primeiros, possuem um manipulador de comunicação ( *"communicator"*) como argumento. Esse comunicador identifica o grupo de processos e o contexto das operações que serão executadas. Os comunicadores proporcionam o mecanismo para identificar um subconjunto de processos, durante o desenvolvimento de programas modulares, assegurando que as mensagens, planejadas para diferentes propósitos, não sejam confundidas.

As funções MPI_INIT e MPI_FINALIZE são usadas, respectivamente, para iniciar e finalizar uma execução MPI. A MPI_INIT deve ser chamada antes de qualquer função MPI e deve ser acionada para cada processador. Depois de ser acionada a MPI_FINALIZE, não se pode acessar outras funções da biblioteca. A função MPI_COMM_SIZE determina o número de processos da execução corrente e a MPI_COMM_RANK os identifica, usando um número inteiro. Os processos, pertencentes a um grupo, são identificados com um número inteiro a partir do número zero. As funções MPI_SEND e MPI_RECV são usadas para enviar e receber mensagens.

Antes de proceder com aspectos mais sofisticados do MPI, é importante assinalar que a programação de troca de mensagens não é determinística, ou seja, a chegada das mensagens enviadas por dois processos A e B ao processo C não é definida. Isto é, o MPI não garante que uma mensagem, enviada de um processo A e de um processo B, chegue na ordem que foi enviada. A responsabilidade disso depende do programador que deve assegurar a execução determinística, quando for solicitada.

A especificação da origem *(source)*, no MPI_RECV, permite o recebimento de mensagens vindas de um único processo específico, ou de qualquer processo. Esta segunda opção permite receber dados que tenha qualquer origem e isso algumas vezes é útil. Porém, a

primeira opção é preferível porque as mensagens chegam na ordem do tempo em que foram enviadas.

Um mecanismo adicional para distinguir as mensagens é através do uso do parâmetro *tag*. O processo de envio deve associar um tag (inteiro) à uma mensagem. O processo de recepção pode especificar que deseja receber mensagens com um tag específico ou com qualquer *anytag*. O uso de um tag é preferível, devido à redução da possibilidade de erros. Nas Tabelas 3 e 4 são mostradas relações de todos os argumentos usados pelas funções MPI_SEND e MPI_RECV, respectivamente.

**SINTAXE EM FORTRAN**: *call* MPI_SEND (sndbuf, count, datatype, dest, tag, comm, mpierr)

Tabela 3: Argumentos da rotina MPI_SEND.

| | |
|---|---|
| **sndbuf** | Endereço inicial do dado que será enviado. Endereço do *aplication buffer* |
| **count** | Número de elementos a serem enviados |
| **datatype** | Tipo do dado |
| **dest** | Identificação do processo destino |
| **tag** | Rótulo da mensagem |
| **comm** | *MPI communicator* |
| **mpierr** | Variável inteira de retorno com o *status* da rotina |

**SINTAXE EM FORTRAN**: *call* MPI_RECV (recvbuf, count, datatype, source, tag, comm, status, mpierr)

Tabela 4: Argumentos da rotina MPI_RECV.

| | |
|---|---|
| **recvbuf** | Variável indicando o endereço do ap*lication buffer* |
| **count** | Número de elementos a serem recebidos |
| **datatype** | Tipo do dado |
| **source** | Identificação da fonte |
| **tag** | Rótulo da mensagem |
| **comm** | *MPI communicator* |
| **status** | Vetor com informações de *source* e *tag* |
| **mpierr** | Variável inteira de retorno com o *status* da rotina |

### 6.5.2 O Ambiente LAM/MPI

LAM *("Local Area Multicomputer")* é uma implementação do MPI baseada em *"daemons"*. Inicialmente, o programa *"lamboot"* cria *"daemons"* LAM baseado na lista de máquinas remotas providenciada pelo usuário. Esses *"daemons"* permanecem parados

nas máquinas remotas até que eles recebam uma mensagem para carregar um arquivo executável MPI para começar a execução. Passos básicos:

- Inicializa-se o LAM, que cria *"daemons"* nas suas máquinas físicas;
- Executa-se programas MPI enquanto os *"daemons"* LAM existem em *"background"*;
- Finalmente, para concluir o LAM, enviam-se comandos de finalização.

No Apêndice A, são apresentados todas as etapas necessárias para o processamento de um programa paralelo no ambiente LAM/MPI.

# 7 ALGORITMOS BUSCA TABU PARALELOS

Neste capítulo serão apresentados os algoritmos de BT paralelo para a resolução do problema estático do planejamento da expansão da transmissão. O objetivo de paralelizar esses algoritmos é aumentar o desempenho (reduzindo o tempo) no processamento, na resolução de grandes problemas matemáticos, como é o caso do PPET, para sistemas de médio e principamente sistemas de grande porte; como também fazer uso de um sistema distribuído para à resolução das tarefas associadas à resolução do problema e também investigar o desempenho dos mesmos em relação à versão seqüencial apresentada anteriormente.

A resolução da função objetivo para o PPET considera-se inicialmente uma população inicial de $n$ configurações. Um algoritmo **BT** seqüencial deve resolver de forma seqüencial, $n$ problemas de programação linear (PPL) para calcular o valor da função objetivo de cada uma das configurações da população. Esse tipo de cálculo ocupa a maior parte do tempo de processamento de um algoritmo seqüencial BT.

Nesta proposta, os processadores são usados para avaliar a função objetivo de subpopulações, parcelas da população total. Isto é feito, dentre outras formas, atribuindo para cada processador o cálculo de um subconjunto de configurações da população total, ou seja, cada filho resolve os $n/nproc$ PPL'$s$ usando $nproc$ processadores e cada uma das $n$ configurações deve ter um valor de função objetivo calculado dos PPL'$s$ pelo programa MINOS e somente serão executados pelos filhos.

Neste trabalho usa-se o modelo de programação distribuída SPMD *("Single Program, Multiple Data")*, que se caracteriza por ter um único programa que processa instâncias em diferentes máquinas; ou seja, uma mesma tarefa é alocada aos diversos processadores sendo que cada nó trabalha com uma porção de dados diferentes do outro. Desta forma, divide-se o espaço de busca da solução conforme a disponibilidade dos processadores.

Os algoritmos BT paralelos propostos são chamados de: *TSP1*, *TSP2*, *TSP3* e foi

desenvolvida uma variante, que é uma extensão da versão 3, chamada *TSP3.B*. Esses algoritmos exploram o espaço de soluções viáveis e buscam a melhor solução ou a solução ótima, através de melhorias da função objetivo. Com o desenvolvimento dos mesmos, busca-se também analisar os seus desempenhos em relação à versão serial, na resolução do PPET em um ambiente multiprocessos.

Na próxima seção será feita uma descrição inicial da função do processo pai e dos processos filhos, para que se tenha uma idéia do mecanismo utilizado na paralelização dos dados; logo a seguir, é mostrado mais detalhadamente o funcionamento dos quatros algoritmos paralelos desenvolvidos. Entretanto, quando da aplicação desta teoria na resolução do PPET, foram implementadas versões paralelas distintas do algoritmo busca tabu de acordo com as seguintes propostas:

- No TSP1, é feita a divisão da população inicial para cada processador e cada processador trabalha com um conjunto de configurações iniciais distintas, com a mesma calibração de parâmetros;
- Na TSP2, é feita a divisão da população inicial para cada processador e cada processador trabalha com um conjunto de configurações iniciais distintas, mas com um conjunto de parâmetros calibrado de forma diferente, atribuindo a cada parâmetro valores distintos para cada processador;
- No TSP3, cada processador trabalha com a mesma população inicial, mas com um conjunto de parâmetros calibrado de forma diferente, atribuindo a cada parâmetro valores distintos para cada processador;
- No TSP3.B, cada processador trabalha com a mesma população inicial, mas a variação de alguns parâmetros foi automatizada. Para que essa variação automática seja feita, é levado em conta o valor que receberá o *"myid"* (número de identificação de cada processador).

## 7.1 Algoritmo TSP1

No TSP1 o pai é encarregado de dividir a população inicial *mpop* em parcelas dadas pela variável *npop/nproc* e enviar aos filhos utilizando-se de primitivas adequadas da biblioteca MPI. Os filhos recebem as parcelas da população inicial dada por *npop/nproc* e calculam a função objetivo dessas configurações. A divisão da população inicial *npop* é feita proporcionalmente ao número de processadores que estão disponíveis para trabalhar.

Se existir um resto, produto da divisão das configurações entre os filhos, este resto será atribuido ao primeiro filho. O número de processadores depende do número de máquinas montadas no ambiente paralelo.

A questão da calibração de parâmetros, foi bastante trabalhada não somente nesta versão, mas em todas as versões desenvolvidas, pelo fato, de que o algoritmo BT é bem complexo de se calibrar. Nesta versão manteve uma única calibração para todos os processadores e foram feitas várias simulações com vários sistemas testes, de pequeno, médio e grande porte. Foram analisados o sistema de Garver com e sem redespacho, o Sul brasileiro com e sem redespacho, o Norte-Nordeste brasileito planos 2002 e 2008.

Na seqüência, é detalhado o algoritmo TSP1 proposto.

**Descrição do Algoritmo TSP1**

**Passo 0:**

1. Montar a máquina paralela virtual com um determinado número de processadores *nproc*;
2. Definir o tamanho da população *(npop)*, determinando previamente as configurações iniciais com o auxílio de um programa inicializador;
3. Se *myid* for igual a 0, então vai ao Passo 1 do processo pai. Caso contrário, vai ao Passo 1, dos processos filhos.

Processo pai

- **Passo 1:** Faz a leitura dos dados iniciais da rede elétrica;
- **Passo 2:** Forma a população inicial;
- **Passo 3:** Faz a divisão da população inicial $linh\_pro = npop/nproc - 1$. OBS: caso haja resto, este será processado pelo primeiro filho;
- **Passo 4:** Envia a cada processo filho a sua parcela correspondente, segundo o Passo 3. OBS: *nproc* é o número total de processadores, incluido o pai. $nproc - 1$, o número de processadores excluindo o pai e que fazem a paralelização segundo a lógica do algoritmo serial;
- **Passo 5:** Pára.

Processos filhos

- **Passo 1:** Recebem as parcelas da população inicial *npop/nproc* e as armazenam;
- **Passo 2:** Calculam o custo das linhas à acrescentar na variável *nsum* e separam suas respectivas configurações na variável $nx$;
- **Passo 3:** Determinam os valores das funções objetivos das *npop/nproc* configurações, usando o algoritmo de PL (MINOS paralelo);
- **Passo 4:** Obtém-se do resultado do fluxo de carga o corte de carga *wk*. Soma-se a ele o custo das linhas *nsum* do passo anterior, e armazenam-se na variável *mobj(kp)*, em que *kp* vai desde um até o número de configurações por filho;
- **Passo 5:** Todas as *npop/nproc* configurações de cada filho passam pelo processo de intensificação;
- **Passo 6:** As melhores configurações são armazenadas e submetidas ao processo de diversificação;
- **Passo 7:** Diversificar por freqüência, *se* $yfl > 0,15$. Caso contrário, diversificar por *"path relinking"*. Em ambos os casos se calcula o valor da F.O. das configurações;
- **Passo 8:** Intensifica novamente;
- **Passo 9:** Critério de parada; pára, se for atingido um máximo número de PPL's resolvidos e vai ao Passo 10; caso contrário, volta ao Passo 2;
- **Passo 10:** Selecionam-se as melhores configurações encontradas;
- **Passo 11:** Imprime os resultados e PÁRA.

Na Figura 7, é mostrado o fluxograma de como funciona o TSP1.

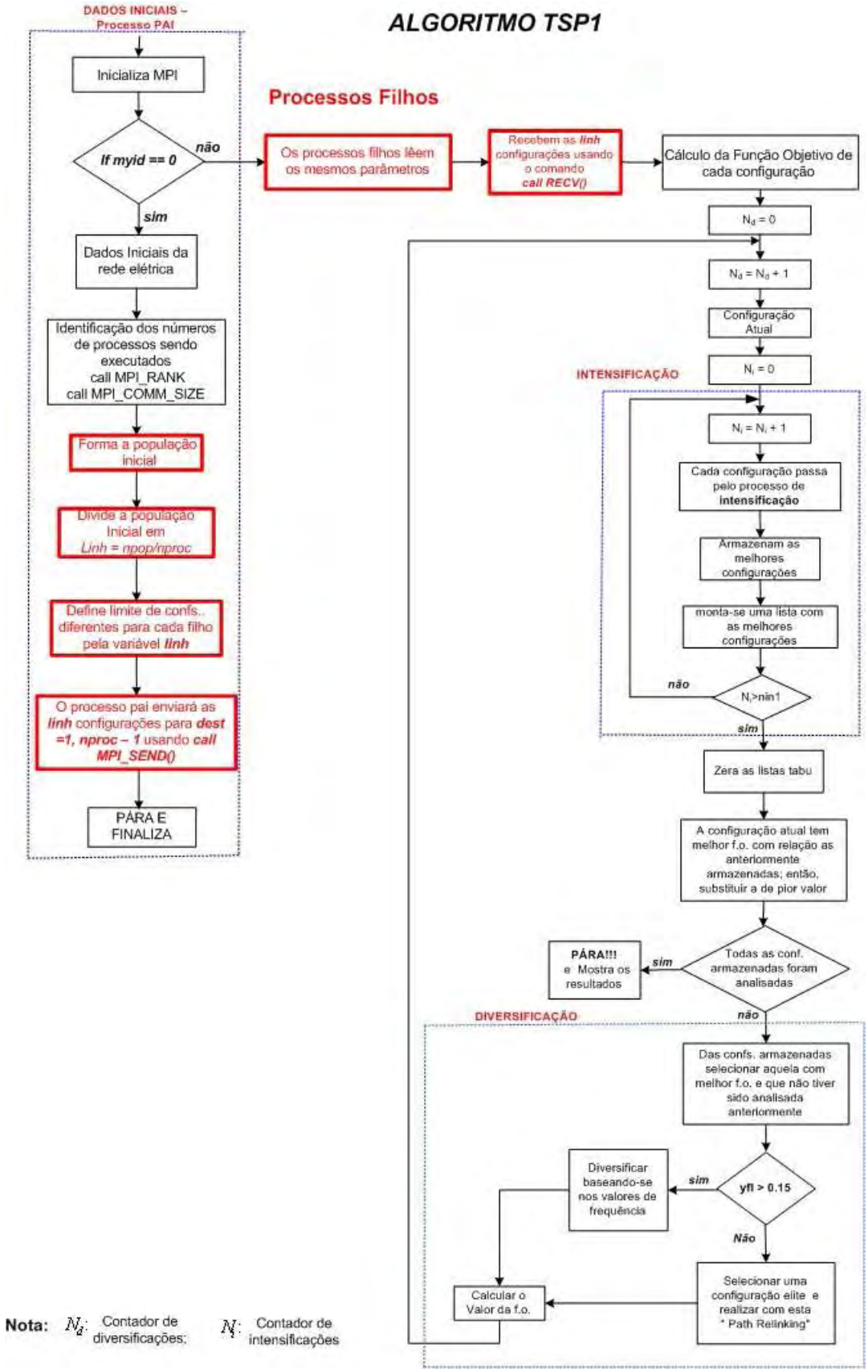



Figura 7: O algoritmo BT paralelo TSP1.

## 7.2 Algoritmo TSP2

Nesta versão, o pai é encarregado de dividir a população inicial *mpop* em parcelas dadas pela variável *npop/nproc* e enviar aos filhos utilizando-se de primitivas adequadas da biblioteca MPI. Os filhos recebem as parcelas da população inicial dada por *npop/nproc*, fazem a leitura dos diferentes parâmetros com diferentes valores e calculam a função objetivo dessas configurações. A divisão da população inicial *npop* é feita proporcionalmente ao número de processadores que estão disponíveis para trabalhar. Se existir um resto, produto da divisão das configurações entre os filhos, este resto será atribuido ao primeiro filho. O número de processadores depende do número de máquinas montadas no ambiente paralelo. A variação dos parâmetros é feita através de arquivos distintos, criados para cada processador, isto possibilita fazer a variação de qualquer parâmetro e alterar seu valor até que o algoritmo mostre um melhor desempenho; após encontrar a melhor calibração, fixam-se os valores encontrados.

**Descrição do Algoritmo TSP2**

**Passo 0:**

1. Montar a máquina paralela virtual com um determinado número de processadores *nproc*;
2. Definir o tamanho da população *(npop)*, determinando previamente as configurações iniciais com o auxílio de um programa inicializador;
3. Se *myid* for igual a 0, então vai ao Passo 1 do processo pai. Caso contrário, vai ao Passo 1, dos processos filhos.

Processo pai

- **Passo 1:** Faz a leitura dos dados iniciais da rede elétrica;
- **Passo 2:** Forma a população inicial;
- **Passo 3:** Faz a divisão da população inicial $linh\_pro = npop/nproc - 1$. OBS: caso haja resto, este será processado pelo primeiro filho;
- **Passo 4:** Envia a cada processo filho a sua parcela correspondente, segundo o Passo 3. OBS: *nproc* é o número total de processadores incluindo o pai. $nproc - 1$ é o número de processadores, excluido o pai, que fazem a paralelização segundo a lógica do algoritmo serial.

Processos filhos

- **Passo 1:** Cada processador lê um conjunto de parâmetros diferentes;
- **Passo 2:** Recebem as parcelas da população inicial *npop/nproc* e as armazenam;
- **Passo 3:** Calculam o custo das linhas a acrescentar na variável *nsum* e separam suas respectivas configurações na variável $nx$;
- **Passo 4:** Determinam os valores das funções objetivos das *npop/nproc* configurações, usando o algoritmo de PL (MINOS paralelo);
- **Passo 5:** Obtém-se do resultado do fluxo de carga o corte de carga *wk*. Soma-se a ele o custo das linhas $nsum$, do passo anterior, e armazena-se na variável *mobj(kp)*, em que $kp$ vai desde um até o número de configurações por filho;
- **Passo 6:** Todas as *npop/nproc* configurações de cada filho passam pelo processo de intensificação;
- **Passo 7:** As melhores configurações são armazenadas e submetidas ao processo de diversificação;
- **Passo 8:** Diversificar por freqüência, *se* $yfl > 0,15$. Caso contrário, diversificar por *"path relinking"*. Em ambos os casos se calcula o valor da F.O. das configurações;
- **Passo 9:** Intensifica novamente;
- **Passo 10:** Critério de parada; pára, se for atingido um máximo número de PPL's resolvidos e vai ao Passo 11; caso contrário, volta ao Passo 3;
- **Passo 11:** Seleciona as melhores configurações encontrada;
- **Passo 11:** Imprime os resultados e PÁRA.

Na Figura 8, é mostrado o fluxograma de como funciona o TSP2.

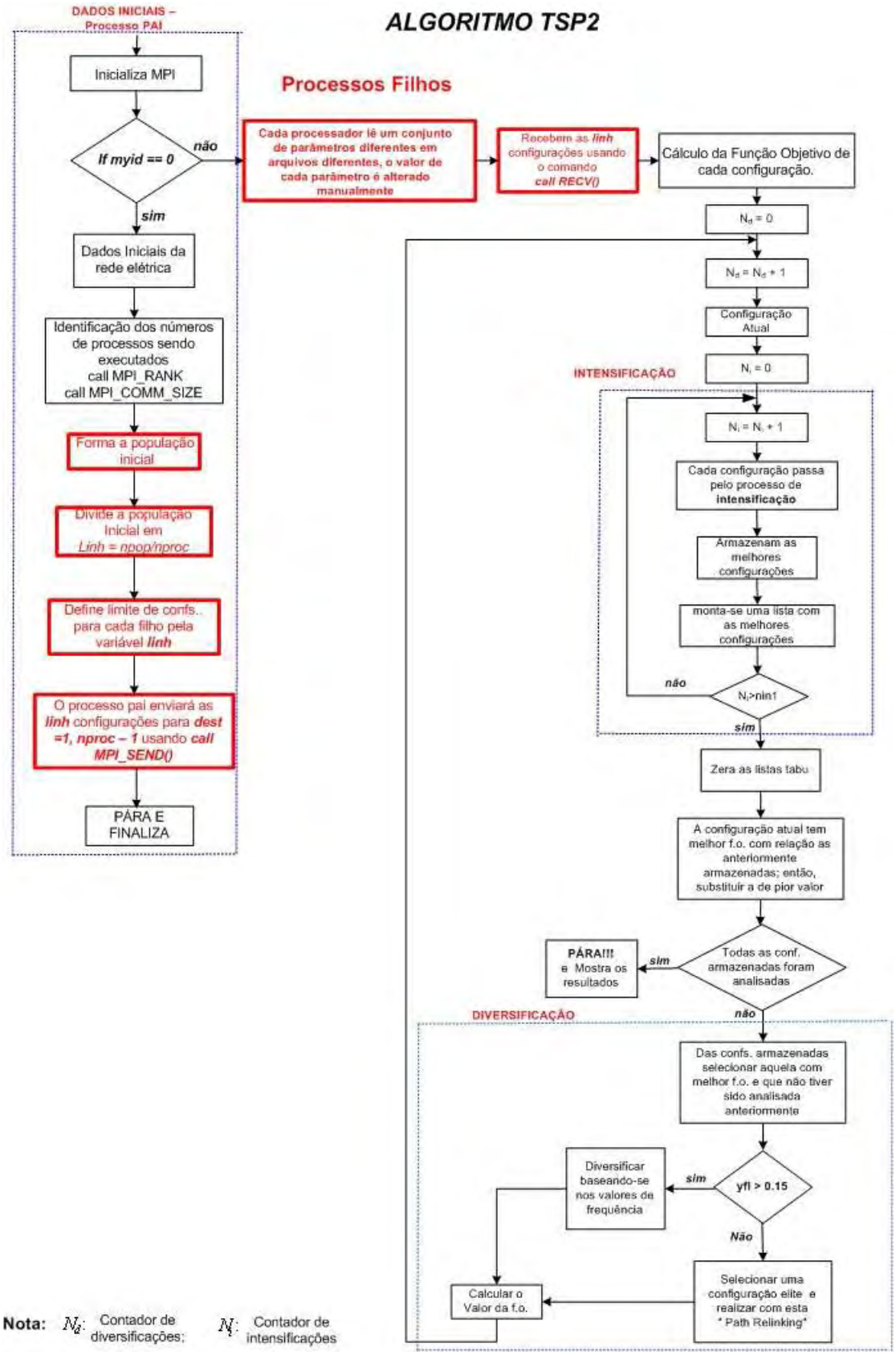



Figura 8: O algoritmo BT paralelo TSP2.

## 7.3 Algoritmo TSP3

No TSP3 todos os processadores trabalham, mas com parâmetros deferenciados e recebem a mesma população inicial dada por *npop*; os mesmos calculam a função objetivo dessas configurações. O número de processadores depende do número de máquinas montadas no ambiente paralelo.

Na seqüência, é detalhado o algoritmo proposto.

**Descrição do Algoritmo TSP3**

**Passo 0:**

1. Montar a máquina paralela virtual com um determinado número de processadores *nproc*;
2. Definir o tamanho da população *(npop)*, determinando previamente as configurações iniciais com o auxílio de um programa inicializador.

Processos Pai e Filhos

- **Passo 1:** Fazem a leitura dos dados iniciais da rede elétrica;
- **Passo 2:** Forma-se a população inicial, com o mesmo número de configurações para cada processador;
- **Passo 3:** Cada processador lê um conjunto de parâmetros diferentes;
- **Passo 4:** Calculam o custo das linhas a acrescentar na variável *nsum* e separam suas respectivas configurações na variável *nx*;
- **Passo 5:** Determinam os valores das funções objetivos das configurações iniciais, usando o algoritmo de PL (MINOS paralelo);
- **Passo 5:** Obtém-se do resultado do fluxo de carga o corte de carga *wk*. Soma-se a ele o custo das linhas *nsum*, do passo anterior, e se armazena-se na variável *mobj(kp)*, em que *kp* vai desde um até o número de configurações por filho;
- **Passo 6:** Todas as configurações de cada processador passam pelo processo de intensificação;

- **Passo 7:** As melhores configurações são armazenadas e submetidas ao processo de diversificação;
- **Passo 8:** Diversificar por freqüência, *se* $yfl > 0,15$. Caso contrário, diversificar por *"path relinking"*. Em ambos os casos se calcula o valor da F.O. das configurações;
- **Passo 9:** Intensifica novamente;
- **Passo 10:** Critério de parada; pára, se for atingido um máximo número de PPL's resolvidos e vai ao Passo 11; caso contrário, volta ao Passo 4;
- **Passo 11:** Cada processador seleciona as melhores configurações encontradas, imprime os resultados e PÁRA.

Na Figura 9, é mostrado o fluxograma de como funciona o TSP3.

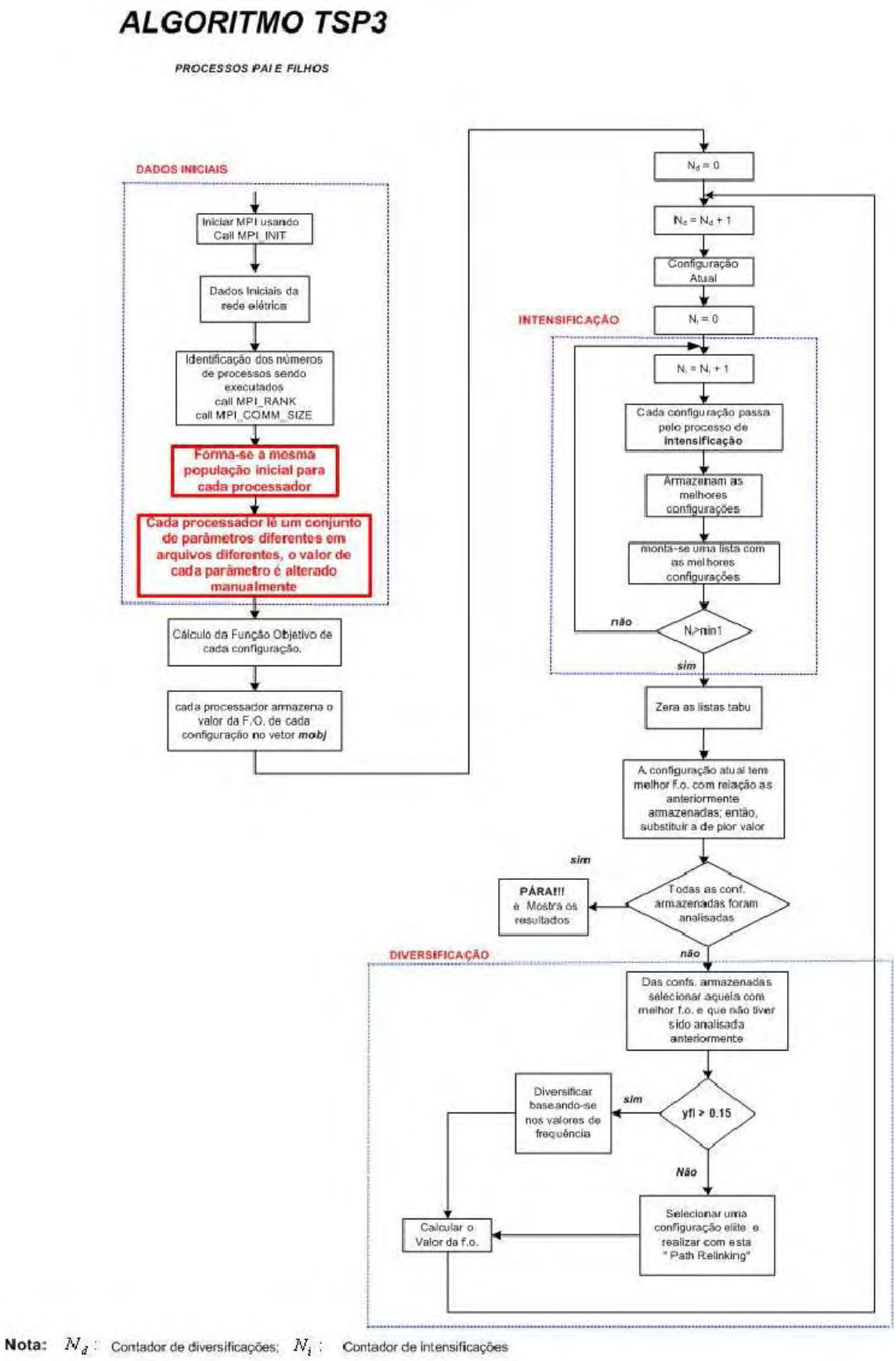



Figura 9: O algoritmo BT paralelo TSP3.

## 7.4 Algoritmo TSP3.B

Esta versão é uma extensão da versão 3, a única diferença é que a variação de alguns parâmetros é feita automaticamente, dependendo o valor que é atribuido ao *myid*. A população inicial é dada por *npop*, e todos os processadores incluido o pai trabalham com a mesma população inicial, e calculam a função objetivo dessas configurações. O número de processadores depende do número de máquinas montadas no ambiente paralelo.

Na seqüência, é detalhado o algoritmo proposto.

**Descrição do Algoritmo TSP3.B**

**Passo 0:**

1. Montar a máquina paralela virtual com um determinado número de processadores *nproc*;
2. Definir o tamanho da população *(npop)*, determinando previamente as configurações iniciais com o auxílio de um programa inicializador.

Processos Pai e Filhos

- **Passo 1:** Fazem a leitura dos dados iniciais da rede elétrica;
- **Passo 2:** Forma-se a população inicial, com o mesmo número de configurações para cada processador;
- **Passo 3:** Cada processador faz a variação automática de determinados parâmetros, baseado na variação do *myid*;
- **Passo 4:** Calculam o custo das linhas a acrescentar na variável *nsum* e separam suas respectivas configurações na variável *nx*;
- **Passo 5:** Determinam os valores das funções objetivos das configurações iniciais, usando o algoritmo de PL (MINOS paralelo);
- **Passo 5:** Obtém-se do resultado do fluxo de carga o corte de carga *wk*. Soma-se a ele o custo das linhas *nsum* do passo anterior e se armazenam na variável *mobj(kp)*, em que *kp* vai desde um até o número de configurações por filho;
- **Passo 6:** Todas as configurações de cada processador passam pelo processo de intensificação;

- **Passo 7:** As melhores configurações são armazenadas e submetidas ao processo de diversificação;
- **Passo 8:** Diversificar por freqüência, *se* $yfl > 0,15$. Caso contrário, diversificar por *path relinking*. Em ambos os casos se calcula o valor da F.O. da configuração;
- **Passo 9:** Intensifica novamente;
- **Passo 10:** Critério de parada; para, se for atingido um máximo número de PPL's resolvidos e vai ao Passo 11; caso contrário, volta ao Passo 4;
- **Passo 11:** Cada processador seleciona as melhores configurações encontradas, imprime os resultados e PÁRA.

Na Figura 10, é mostrado o fluxograma de como funciona o TSP3B.

A calibração automática de alguns dos parâmetros é feita por exemplo, no caso da variável *ncorte*, somando-se com o valor da variável *myid*. Esta calibração é para sistema de pequeno porte, como o sistema Garver. Para sistema de médio porte, como o Sul brasileiro, a calibração é feita subtraindo-se do valor de *ncorte* duas vezes o valor de *myid*. Para sistemas maiores, como o Norte-Nordeste brasileiro, a variável *ncorte* é multiplicada pelo valor que receber a variável *myid* e subtraido cinqüenta unidades. Com isso, para cada processador esta variável terá valores distintos. A escolha de como variar determinada variável é feita também de acordo com o sistema teste analisado, ou seja, para sistema de pequeno porte, médio e grande porte a automação é feita de forma distinta.

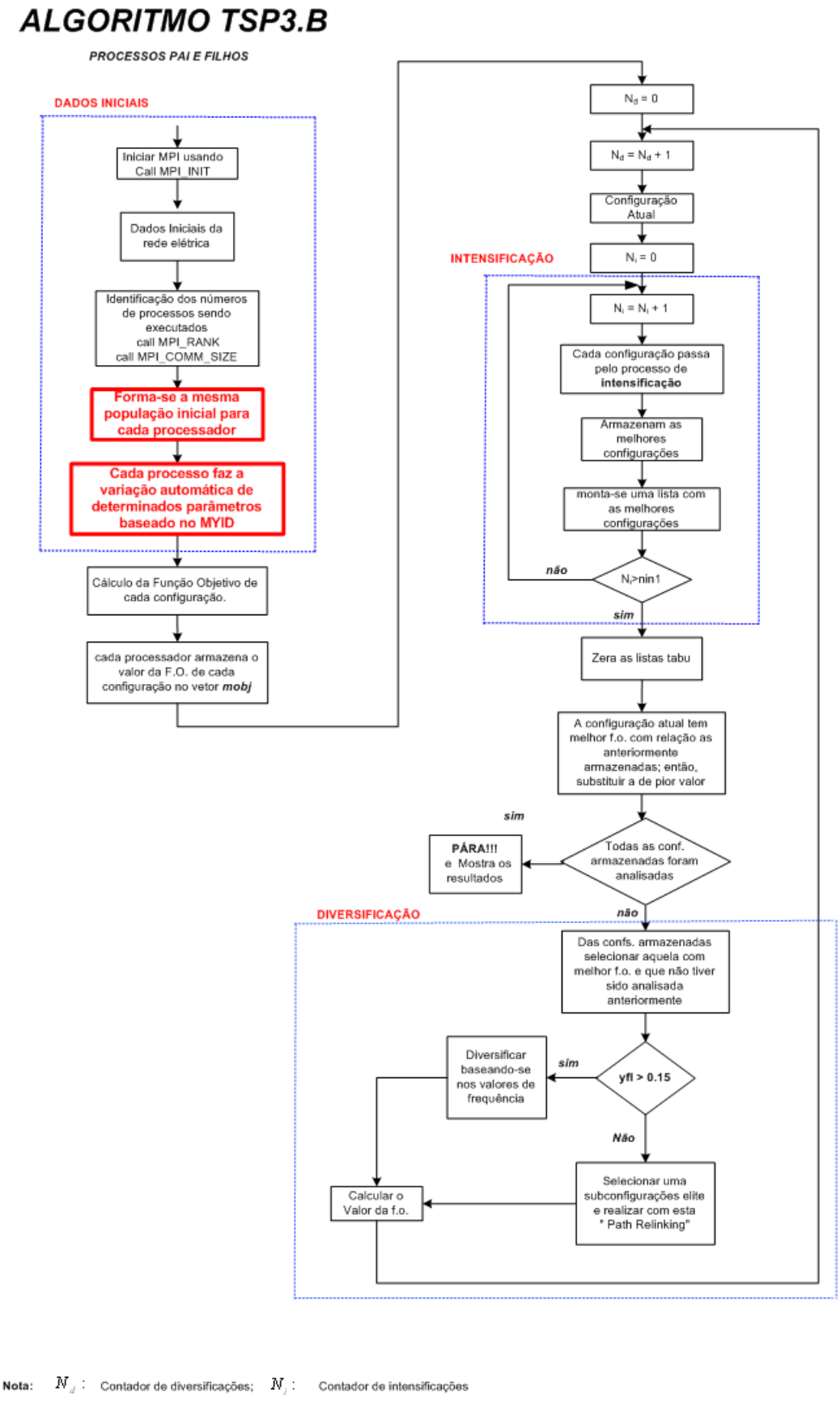



Figura 10: O algoritmo BT paralelo TSP3.B.

# 8 TESTES E RESULTADOS COM SISTEMAS DE PEQUENO, MÉDIO E GRANDE PORTE

O algoritmo serial usado neste estudo foi desenvolvido originalmente em Fortran para máquinas SUN (Sparc Ultra1) com sistema operacional Solaris (GALLEGO; DEOLIVEIRA, 1997. 6 p.). Para a realização deste trabalho precisou-se inicialmente adaptar o algoritmo serial para rodar nas máquinas Linux (Fedora Core 5.0), que são as máquinas que compõe o LLPP (Laboratório de Linux e Processamento Paralelo). Foram feitas diversas mudanças e muitos testes, para que o algoritmo funcionasse adequadamente.

Os resultados encontrados com simulações do algoritmo TSNOR, permitem estabelecer o desempenho da metaheurística Busca Tabu quando é usado na resolução do PPET de sistemas de pequeno, médio e grande porte. Também realizou-se uma comparação da BT quando utilizado uma ou várias configurações, assim como o impacto produzido por usar configurações geradas com o algoritmo inicializador. Também é feita uma comparação entre os resultados do TSNOR e as versões paralelas desenvolvidas TSP1, TSP2 e TSP3.

Os parâmetros utilizados na implementação do algoritmo foram calibrados, depois de muitas simulações realizadas, para permitir desempenhos com maior eficiência e qualidade dos resultados em encontrar soluções ótimas dos sistemas de Garver e Sul brasileiro apresentados, além de se procurar obter as melhores soluções possíveis para o sistema Norte-Nordeste brasileiro. Neste trabalho a comparação é feita em relação ao número de ciclos e número de PPL's resolvidos em que o sistema encontrou a solução ótima ou a melhor solução e ao esforço computacional (tempo de processamento), requeridos para cada simulação.

São apresentados resultados de testes realizados com configurações iniciais obtidas via

inicializador (INIC), com todos os sistemas: Garver (com e sem redespacho), Sul brasileiro (com e sem redespacho) e Norte-Nordeste brasileiro (plano 2008 e 2002). Os resultados obtidos, os parâmetros e o desempenho do algoritmo proposto são descritos nas seções que se seguem.

Nas simulações dos algoritmos os parâmetros foram calibrados para se chegar à solução ótima do PPET sempre procurando obter-se o ótimo global (ou a melhor solução - ótima local) no menor tempo possível. Foram feitos diversos testes variando os parâmetros, e observado a variação do número de PPL's a serem resolvidos (e o tempo correspondente) para chegar à convergência. Observa-se a grande dependência nos desempenhos dos algoritmos na obtenção da solução ótima, em relação aos valores dos parâmetros. Então os valores que recebem os parâmetros são críticos no desempenho do algoritmo em encontrar uma solução ótima ou de boa qualidade.

A nomenclatura dos parâmetros usados em todos os sistemas testados pode ser vista na lista de abreviaturas, variáveis e símbolos.

Os dados dos sistemas testes utilizados são apresentados no Apêndice B.

## 8.1 Sistema Garver

O sistema teste Garver é composto por 6 barras e 15 ramos, e para as simulações são consideradas neste sistema as situações de com e sem redespacho conforme dados apresentados no Apêndice B.

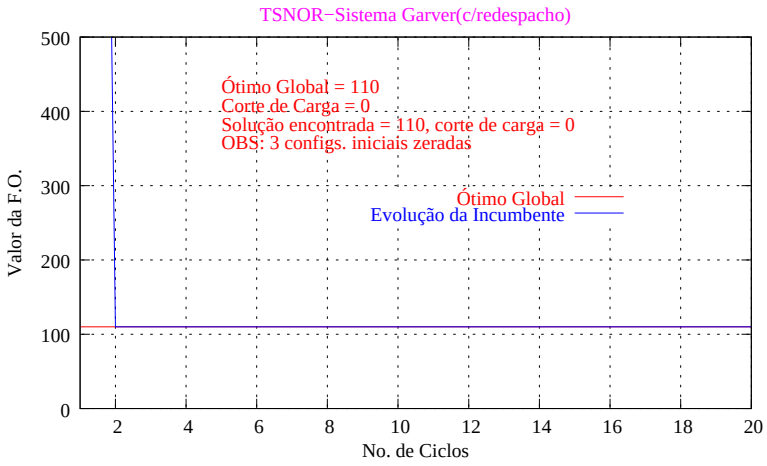
### 8.1.1 Algoritmo TSNOR

O algoritmo TSNOR considerado a seguir é o algoritmo detalhado na seção 5.5, página 46.

#### 8.1.1.1 Sistema Garver (com redespacho)

Na Tabela 5, mostra-se a solução ótima global e a solução encontrada com o algoritmo TSNOR com as configurações iniciais zeradas, para o sistema Garver com redespacho. A solução ótima global foi encontrada com o valor da função objetivo igual a 110, sem corte de carga, em 306 PPLs no 2º ciclo, e o tempo computacional para encontrar a resposta ótima global foi de 6,4 s.

Tabela 5: Sistema Garver - algoritmo TSNOR (com redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 |

**b)** configuração ótima;

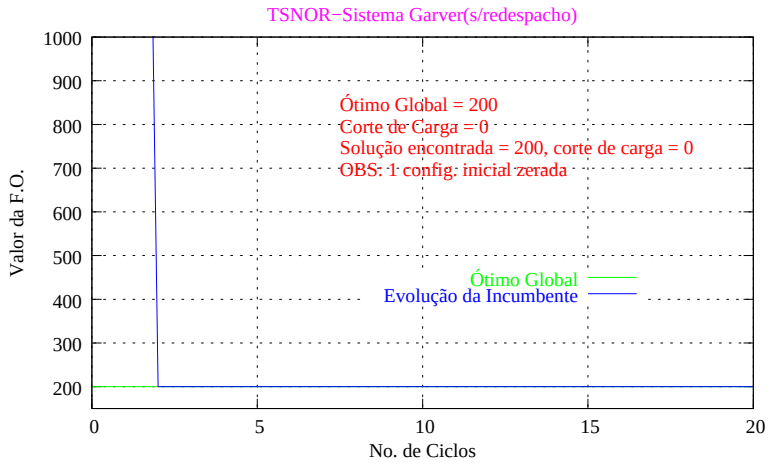
| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | nsaen | 3 |
| ncorte | 0 | ncosto | 0 |
| nintens | 3 | nsem | 3 |
| ndiver1 | 2 | nsem2 | 4 |
| ndiver2 | 3 | melhor | 2 |
| nanaliza | 10 | ntrocas | 4 |
| nduramin | 4 | mtrosi | 1 |
| nduravar | 4 | mtrono | 1 |
| msens | 0 | md1 | 500 |
| mmin1 | 0 | md2 | 1000 |
| mmin2 | 0 | kseed | 3667 |
| mgarv | 1 | nmax | 4 |
| mgera | 0 | nactmax2 | 1 |
| mcarg | 0 | nplmax | 1000 |
| malea | 1 | ncl | 0 |

**c)** parâmetros.

### 8.1.1.2 Sistema Garver (sem redespacho)

Na Tabela 6 são apresentados os dados da simulação com o sistema Garver sem redespacho, com o algoritmo TSNOR. O teste foi realizado com uma configuração inicial zerada. A solução ótima obtida foi de 200, sem corte de carga, em 4,1 s, no 3º ciclo, com 194 PPLs. Pode-se ver mais detalhadamente a evolução do algoritmo no gráfico abaixo.

Tabela 6: Sistema Garver - algoritmo TSNOR (sem redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 0 | 2 | 0 |

**b)** configuração ótima;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | nsaen | 3 |
| ncorte | 0 | ncosto | 0 |
| nintens | 3 | nsem | 1 |
| ndiver1 | 1 | nsem2 | 8 |
| ndiver2 | 3 | melhor | 2 |
| nanaliza | 10 | ntrocas | 4 |
| nduramin | 4 | mtrosi | 1 |
| nduravar | 4 | mtrono | 1 |
| msens | 1 | md1 | 500 |
| mmin1 | 0 | md2 | 1000 |
| mmin2 | 0 | kseed | 3667 |
| mgarv | 1 | nmax | 4 |
| mgera | 0 | nactmax2 | 2 |
| mcarg | 0 | nplmax | 1000 |
| malea | 1 | ncl | 0 |

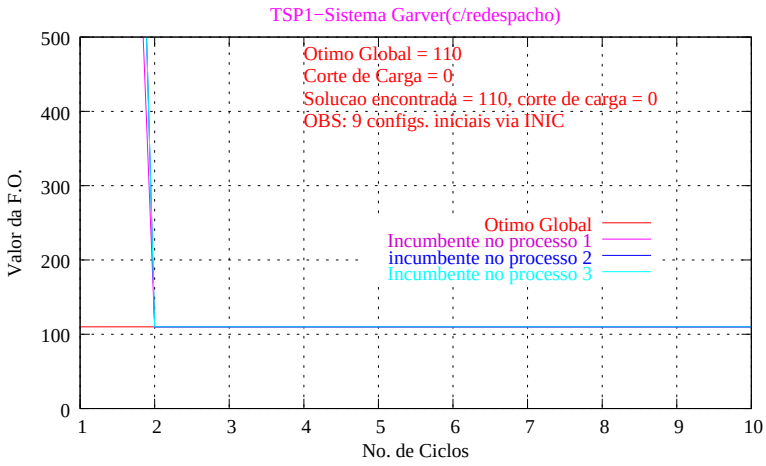
**c)** parâmetros.

## 8.1.2 Algoritmo TSP1

### 8.1.2.1 Sistema Garver (com redespacho)

Para o algoritmo TSP1, com inicialização através do incializador INIC, a solução ótima para o sistema Garver com redespacho foi obtida no 2º ciclo, com 224 PPLs, em um tempo de 0,44 s. Na Tabela 7, estão todos os dados da simulação.

Tabela 7: Sistema Garver - algoritmo TSP1 (com redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 |

**b)** configuração ótima;

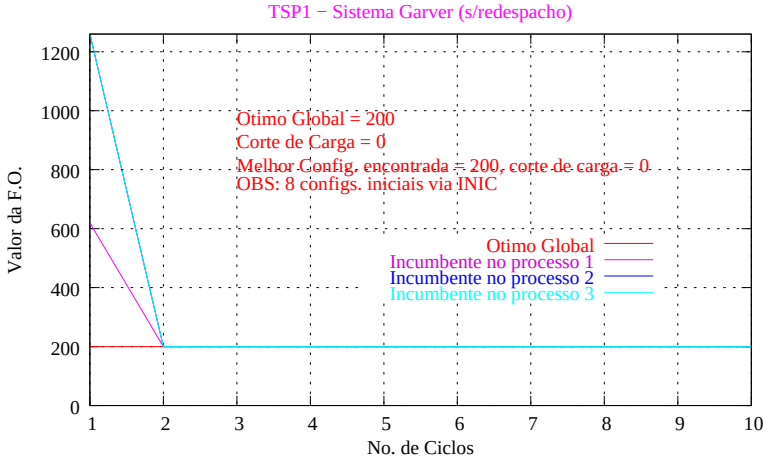
| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | ncosto | 0 |
| ncorte | 0 | nsem | 9 |
| nintens | 3 | nsem2 | 4 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 3 | ntrocas | 7 |
| nanaliza | 10 | mtrosi | 1 |
| nduramin | 4 | mtrono | 1 |
| nduravar | 4 | md1 | 500 |
| mmin1 | 1 | md2 | 1000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 12 | nactmax2 | 14 |
| mcarg | 1 | nplmax | 2000 |
| malea | 1 | ncl | 0 |
| nsaen | 3 | | |

**c)** parâmetros.

#### 8.1.2.2 Sistema Garver (sem redespacho)

Na Tabela 8 são apresentados os dados da simulação com o sistema de Garver sem redespacho, para o algoritmo TSP1. O teste foi realizado com oito configurações iniciais (obtidas via algoritmo de inicialização INIC). A solução ótima obtida foi de 200, sem corte de carga, em 0,11 s, no 2º ciclo, com 51 PPLs. Pode-se ver mais detalhadamente a evolução do algoritmo no gráfico da Tabela 8.

Tabela 8: Sistema Garver - algoritmo TSP1 (sem redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 0 | 2 | 0 |

**b)** configuração ótima;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | ncosto | 0 |
| ncorte | 0 | nsem | 8 |
| nintens | 3 | nsem2 | 5 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 3 | ntrocas | 4 |
| nanaliza | 10 | mtrosi | 1 |
| nduramin | 4 | mtrono | 1 |
| nduravar | 4 | md1 | 500 |
| mmin1 | 0 | md2 | 1000 |
| mmin2 | 0 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 0 | nactmax2 | 1 |
| mcarg | 0 | nplmax | 1000 |
| malea | 0 | ncl | 0 |
| nsaen | 3 | | |

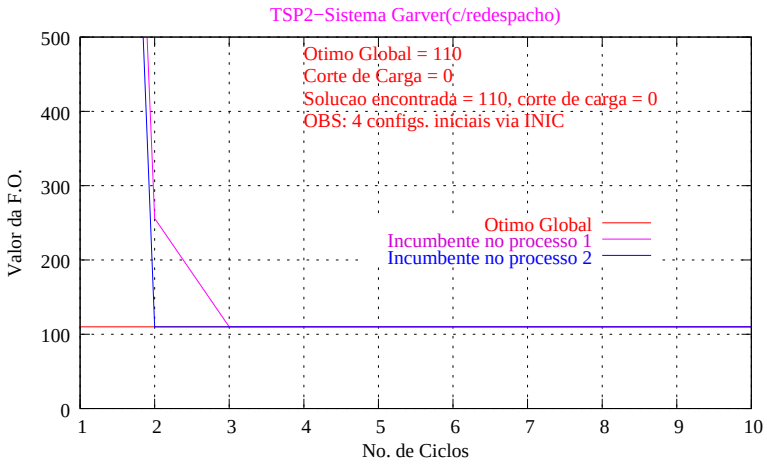
**c)** parâmetros.

## 8.1.3 Algoritmo TSP2

### 8.1.3.1 Sistema Garver (com redespacho)

Para o algoritmo TSP2, com quatro configurações iniciais (obtida via algoritmo de inicialização INIC), a solução ótima para o sistema Garver com redespacho foi obtida no 2º ciclo, com 143 PPLs, em um tempo de 0,47 s. Na Tabela 9, estão todos os dados da simulação.

Tabela 9: Sistema Garver - algoritmo TSP2 (com redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 |

**b)** configuração ótima;

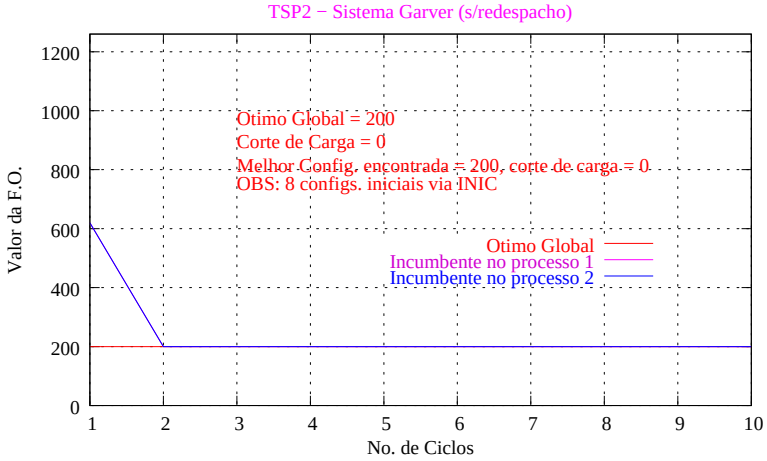
| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | ncosto | 0 |
| ncorte | 0 | nsem | 4 |
| nintens | 3 | nsem2 | 3 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 3 | ntrocas | 7 |
| nanaliza | 10 | mtrosi | 1 |
| nduramin | 4 | mtrono | 1 |
| nduravar | 4 | md1 | 500 |
| mmin1 | 0 | md2 | 1000 |
| mmin2 | 0 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 0 | nactmax2 | 1 |
| mcarg | 0 | nplmax | 2000 |
| malea | 1 | ncl | 0 |
| nsaen | 3 | | |

**c)** parâmetros.

#### 8.1.3.2 Sistema Garver (sem redespacho)

Na Tabela 10 são apresentados os dados da simulação com o sistema de Garver sem redespacho para o algoritmo TSP2. O teste foi realizado com oito configurações iniciais (obtida via algoritmo de inicialização INIC). A solução ótima obtida foi de 200, sem corte de carga, em 0,31 s, no 2º ciclo, com 135 PPLs. Pode-se ver mais detalhadamente a evolução do algoritmo no gráfico da Tabela 10.

Tabela 10: Sistema Garver - algoritmo TSP2 (sem redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 0 | 2 | 0 |

**b)** configuração ótima;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | ncosto | 0 |
| ncorte | 0 | nsem | 8 |
| nintens | 3 | nsem2 | 5 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 3 | ntrocas | 4 |
| nanaliza | 10 | mtrosi | 1 |
| nduramin | 4 | mtrono | 1 |
| nduravar | 4 | md1 | 500 |
| mmin1 | 1 | md2 | 1000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 0 | nactmax2 | 1 |
| mcarg | 0 | nplmax | 1000 |
| malea | 0 | ncl | 0 |
| nsaen | 3 | | |

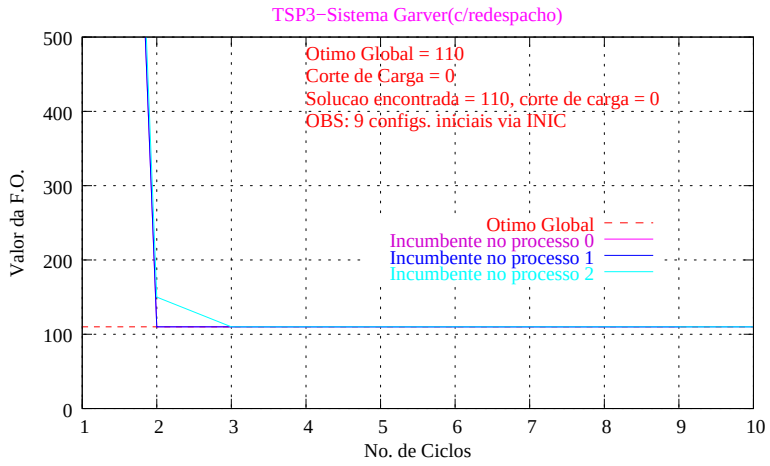
**c)** parâmetros.

## 8.1.4 Algoritmo TSP3

### 8.1.4.1 Sistema Garver (com redespacho)

Para o algoritmo TSP3, com nove configurações iniciais (obtida via algoritmo de inicialização INIC), a solução ótima para o sistema Garver com redespacho foi obtida no 2º ciclo, com 641 PPLs, em um tempo de 1,0 s. Na Tabela 11, estão todos os dados da simulação.

Tabela 11: Sistema Garver - algoritmo TSP3 (com redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 0 |

**b)** configuração ótima;

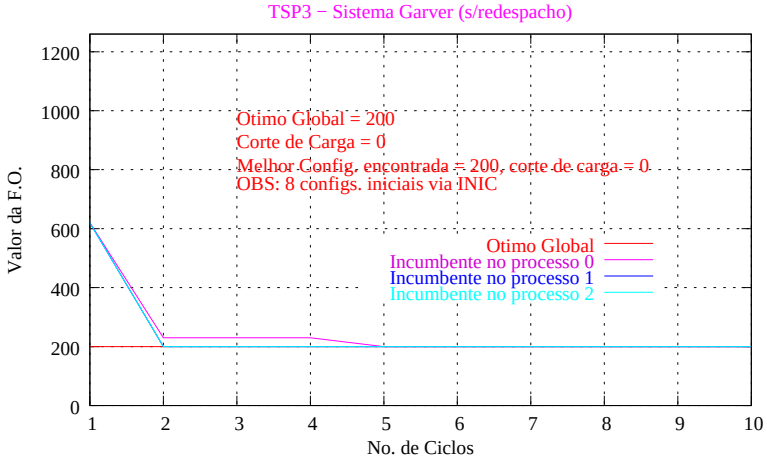
| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | ncosto | 0 |
| ncorte | 0 | nsem | 9 |
| nintens | 3 | nsem2 | 3 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 3 | ntrocas | 7 |
| nanaliza | 10 | mtrosi | 1 |
| nduramin | 4 | mtrono | 1 |
| nduravar | 4 | md1 | 500 |
| mmin1 | 1 | md2 | 1000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 2000 |
| malea | 1 | ncl | 0 |
| nsaen | 3 | | |

**c)** parâmetros.

### 8.1.4.2 Sistema Garver (sem redespacho)

Na Tabela 12 são apresentados os dados da simulação com o sistema de Garver sem redespacho para o algoritmo TSP3. O teste foi realizado com oito configurações iniciais (obtida via algoritmo de inicialização INIC). A solução ótima obtida foi de 200, sem corte de carga, em 0,71 s, no 2º ciclo, com 255 PPLs.

Tabela 12: Sistema Garver - algoritmo TSP3 (sem redespacho).




**a)** gráfico;

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 1 | 0 | 0 | 2 | 0 |

**b)** configuração ótima;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 10 | ncosto | 0 |
| ncorte | 0 | nsem | 8 |
| nintens | 3 | nsem2 | 5 |
| ndiver1 | 2 | melhor | 2 |
| ndiver2 | 3 | ntrocas | 7 |
| nanaliza | 10 | mtrosi | 1 |
| nduramin | 4 | mtrono | 1 |
| nduravar | 4 | md1 | 500 |
| mmin1 | 0 | md2 | 1000 |
| mmin2 | 0 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 0 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 1000 |
| malea | 0 | ncl | 0 |
| nsaen | 3 | | |

**c)** parâmetros.

## 8.2 Sistema Sul Brasileiro

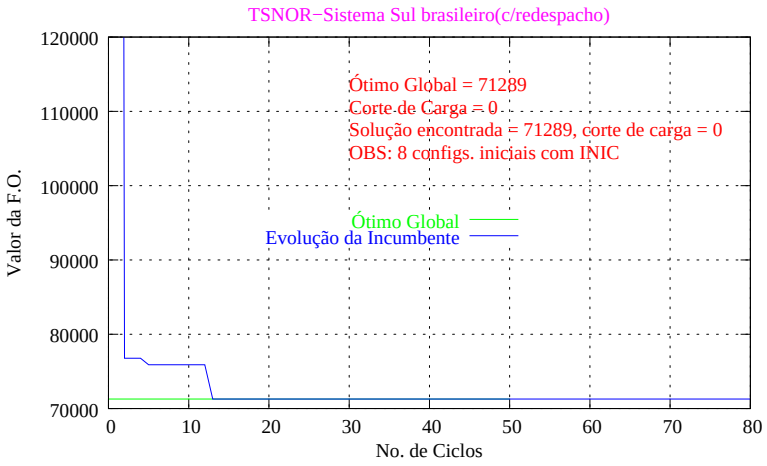
O sistema teste Sul brasileiro é composto por 79 linhas e 46 barras, e para as simulações com este sistema são consideradas as situações de com e sem redespacho, conforme dados apresentados no Apêndice B.

### 8.2.1 Algoritmo TSNOR

#### 8.2.1.1 Sistema Sul (com redespacho)

Na melhor simulação do algoritmo TSNOR para o sistema Sul brasileiro com redespacho, a solução ótima foi encontrada no 13º ciclo, com o valor de US$ 71,289,000.00, sem corte de carga, e foram resolvidos 1.461 PPLs em 46,1 s. Nessa simulação, utilizou-se oito configurações iniciais obtida via algoritmo INIC. Os dados da simulação são apresentados na Tabela 13.

Tabela 13: Sistema Sul Brasileiro - algoritmo TSNOR (com redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | nsaen | 7 |
| ncorte | 0 | ncosto | 0 |
| nintens | 4 | nsem | 8 |
| ndiver1 | 1 | nsem2 | 50 |
| ndiver2 | 4 | melhor | 2 |
| nanaliza | 100 | ntrocas | 12 |
| nduramin | 9 | mtrosi | 1 |
| nduravar | 9 | mtrono | 1 |
| msens | 1 | md1 | 230000 |
| mmin1 | 0 | md2 | 45000 |
| mmin2 | 0 | kseed | 3667 |
| mgarv | 3 | nmax | 4 |
| mgera | 0 | nactmax2 | 1 |
| mcarg | 0 | nplmax | 1000 |
| malea | 1 | ncl | 100 |

**c)** parâmetros.

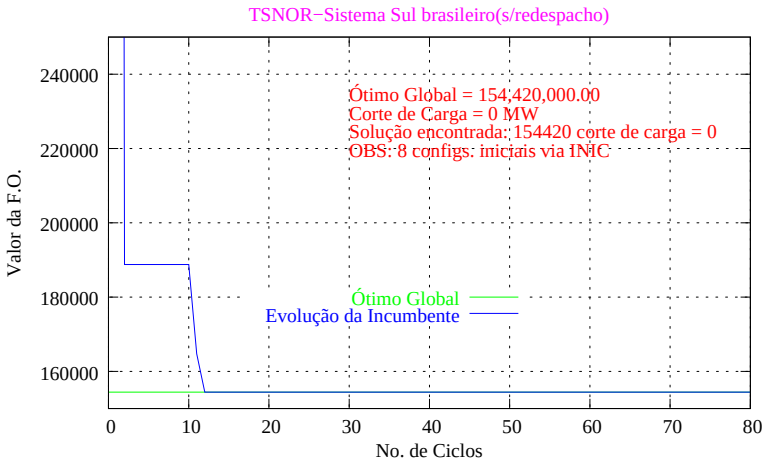
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | |

**b)** configuração ótima;

#### 8.2.1.2 Sistema Sul (sem redespacho)

A configuração ótima global e todos os dados da simulação obtidos com o algoritmo TSNOR para o sistema Sul brasileiro sem redespacho estão na Tabela 14; sendo que são representados no vetor de configuração ótima apenas os ramos que foram acrescentadas linhas. A solução ótima encontrada foi de US$ 154,420,000.00, sem corte de carga, com 1.868 PPLs no 12º ciclo, com o tempo de 63,2 s. Na Tabela 14 apresentam-se os dados desta simulação, efetuada com oito configurações iniciais abtidas via algoritmo inicializador INIC.

Tabela 14: Sistema Sul Brasileiro - algoritmo TSNOR (sem redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | nsaen | 7 |
| ncorte | 0 | ncosto | 0 |
| nintens | 4 | nsem | 8 |
| ndiver1 | 4 | nsem2 | 50 |
| ndiver2 | 4 | melhor | 2 |
| nanaliza | 100 | ntrocas | 12 |
| nduramin | 9 | mtrosi | 1 |
| nduravar | 9 | mtrono | 1 |
| msens | 2 | md1 | 230000 |
| mmin1 | 1 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 3 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 10000 |
| malea | 1 | ncl | 10 |

**c)** parâmetros.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | |

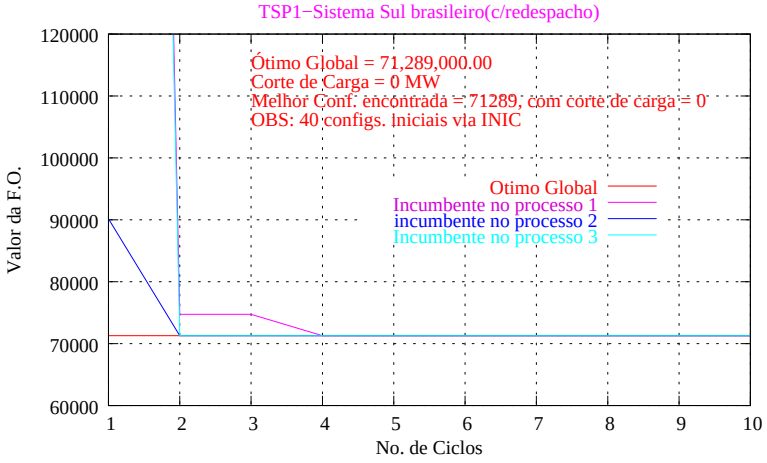
**b)** configuração ótima;

## 8.2.2 Algoritmo TSP1

### 8.2.2.1 Sistema Sul (com redespacho)

Com este algoritmo foram feitos diversos testes. Na melhor simulação o algoritmo TSP1 para o sistema Sul brasileiro com redespacho a solução ótima foi encontrada no 4º ciclo, com o valor de US$ 71,289,000.00, sem corte de carga, e foram resolvidos 873 PPLs em 6,3 s. Os dados da simulação são apresentados na Tabela 15, onde utilizou-se quarenta configurações iniciais geradas previamente com o inicializador INIC.

Tabela 15: Sistema Sul Brasileiro - algoritmo TSP1 (com redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 0 |
| ncorte | 0 | nsem | 40 |
| nintens | 4 | nsem2 | 70 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 4 | ntrocas | 12 |
| nanaliza | 100 | mtrosi | 1 |
| nduramin | 9 | mtrono | 1 |
| nduravar | 9 | md1 | 230000 |
| mmin1 | 1 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 3 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 3 | nplmax | 10000 |
| malea | 0 | ncl | 10 |
| nsaen | 7 | | |

**c)** parâmetros.

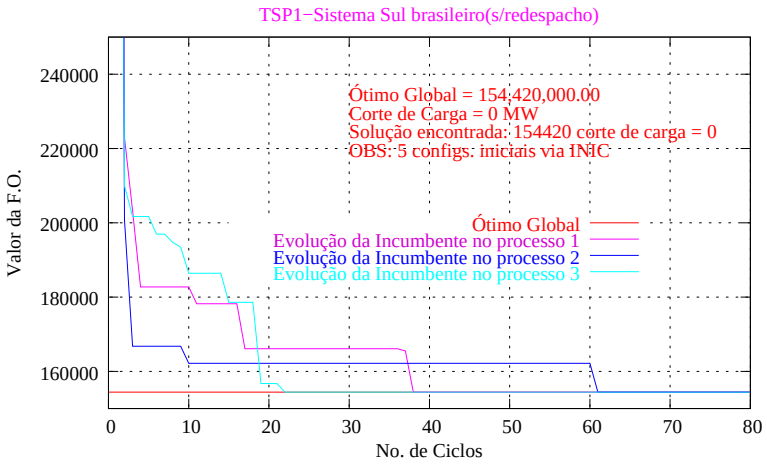
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | |

**b)** configuração ótima;

### 8.2.2.2 Sistema Sul (sem redespacho)

A configuração ótima global e todos os dados da simulação obtidos com o algoritmo TSP1 para o sistema Sul brasileiro sem redespacho estão na Tabela 16; sendo que são apresentados na configuração ótima apenas os ramos que foram acrescentadas linhas. A solução ótima encontrada foi de US$ 154,420,000.00, sem corte de carga, com 1.955 PPLs no 22º ciclo, com o tempo de 15 s. Na Tabela 16 apresentam-se os dados desta simulação, realizada com cinco configurações iniciais obtidas via inicializador INIC.

Tabela 16: Sistema Sul Brasileiro - algoritmo TSP1 (sem redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 0 |
| ncorte | 0 | nsem | 5 |
| nintens | 4 | nsem2 | 50 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 4 | ntrocas | 12 |
| nanaliza | 100 | mtrosi | 1 |
| nduramin | 9 | mtrono | 1 |
| nduravar | 9 | md1 | 230000 |
| mmin1 | 3 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 10000 |
| malea | 6 | ncl | 10 |
| nsaen | 7 | | |

**c)** parâmetros.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | |

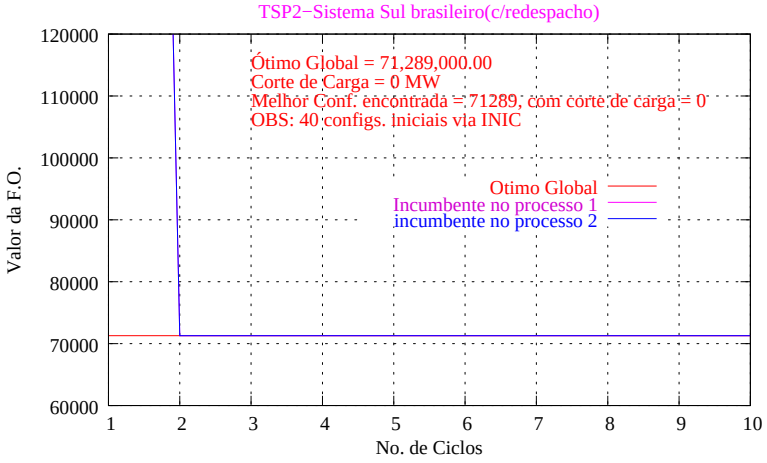
**b)** configuração ótima;

## 8.2.3 Algoritmo TSP2

### 8.2.3.1 Sistema Sul (com redespacho)

Com o algoritmo TSP2 foram feitos diversos testes, para o sistema Sul brasileiro com redespacho, e a melhor simulação realizada encontrou a solução ótima no 2º ciclo, com o valor de US$ 71,289,000.00, sem corte de carga, com 966 PPLs resolvidos em 7,7 s. Nesta simulação utilizou-se quarenta configurações iniciais obtidas com o algoritmo INIC. Os dados da simulação são apresentados na Tabela 17.

Tabela 17: Sistema Sul Brasileiro - algoritmo TSP2 (com redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 0 |
| ncorte | 0 | nsem | 40 |
| nintens | 4 | nsem2 | 50 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 4 | ntrocas | 12 |
| nanaliza | 100 | mtrosi | 1 |
| nduramin | 9 | mtrono | 1 |
| nduravar | 9 | md1 | 230000 |
| mmin1 | 1 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 1 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 8 | nplmax | 10000 |
| malea | 0 | ncl | 10 |
| nsaen | 7 | | |

**c)** parâmetros.

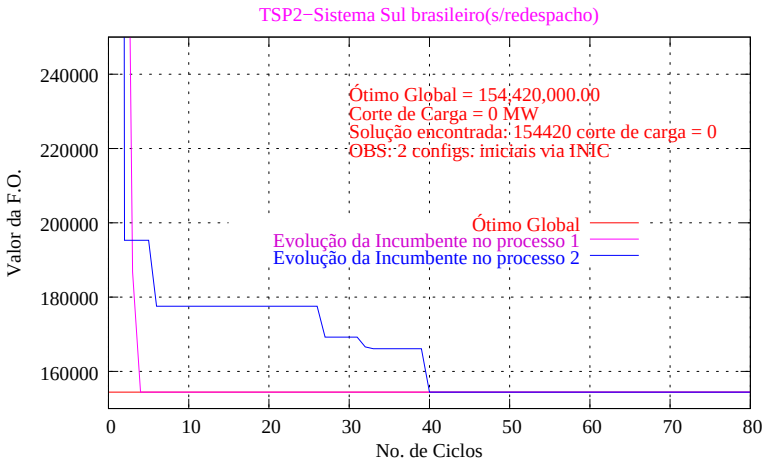
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | |

**b)** configuração ótima;

### 8.2.3.2 Sistema Sul (sem redespacho)

Na Tabela 18, são apresentados a configuração ótima global e todos os dados da simulação realizados com duas configurações iniciais obtidas via INIC, para o sistema Sul brasileiro sem redespacho. A solução ótima encontrada foi de US$ 154,420,000.00, sem corte de carga, com 581 PPLs no 4º ciclo, com o tempo de 4,4 s.

Tabela 18: Sistema Sul Brasileiro - algoritmo TSP2 (sem redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 0 |
| ncorte | 0 | nsem | 2 |
| nintens | 4 | nsem2 | 50 |
| ndiver1 | 4 | melhor | 2 |
| ndiver2 | 4 | ntrocas | 12 |
| nanaliza | 100 | mtrosi | 1 |
| nduramin | 9 | mtrono | 1 |
| nduravar | 9 | md1 | 230000 |
| mmin1 | 3 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 1 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 10000 |
| malea | 6 | ncl | 10 |
| nsaen | 7 | | |

**c)** parâmetros.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | |

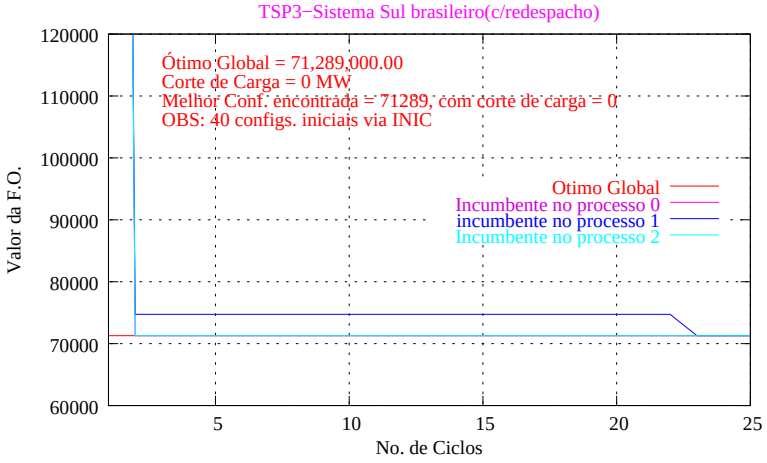
**b)** configuração ótima;

## 8.2.4 Algoritmo TSP3

### 8.2.4.1 Sistema Sul (com redespacho)

Com este algoritmo foram feitos diversos testes para o sistema Sul brasileiro com redespacho e na melhor simulação a solução ótima foi encontrada no 2º ciclo, com o valor de US$ 71,289,000.00, sem corte de carga, e foram resolvidos 2.305 PPLs em 16,3 s. Os dados desta simulação, realizada com quarenta configurações iniciais obtidas com o algoritmo INIC, são apresentados na Tabela 19.

Tabela 19: Sistema Sul Brasileiro - algoritmo TSP3 (com redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 0 |
| ncorte | 0 | nsem | 40 |
| nintens | 4 | nsem2 | 50 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 4 | ntrocas | 10 |
| nanaliza | 100 | mtrosi | 1 |
| nduramin | 9 | mtrono | 1 |
| nduravar | 9 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 2 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 10000 |
| malea | 8 | ncl | 10 |
| nsaen | 7 | | |

**c)** parâmetros.

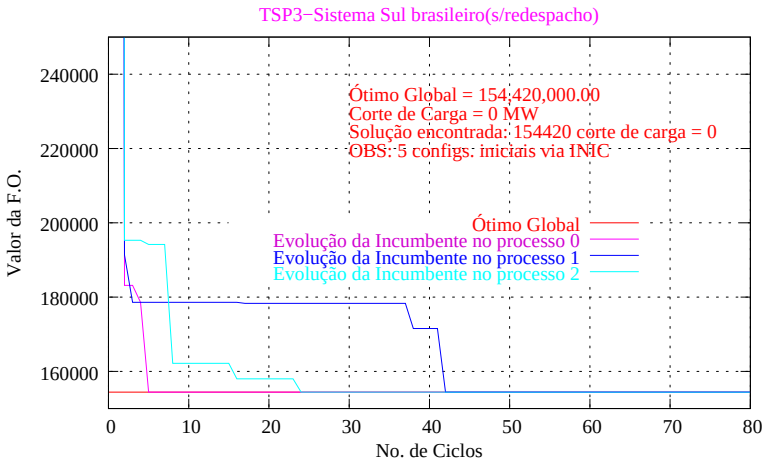
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | |

**b)** configuração ótima;

### 8.2.4.2 Sistema Sul (sem redespacho)

A configuração ótima global e todos os dados da simulação para os sistema Sul brasileiro sem redespacho, com o algoritmo TSP3, usando cinco configurações iniciais via INIC, estão na Tabela 20. A solução ótima encontrada foi de US$ 154,420,000.00, sem corte de carga, com 794 PPLs no 5º ciclo, com o tempo de 5,9 s.

Tabela 20: Sistema Sul Brasileiro - algoritmo TSP3 (sem redespacho).




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 0 |
| ncorte | 0 | nsem | 5 |
| nintens | 4 | nsem2 | 50 |
| ndiver1 | 2 | melhor | 2 |
| ndiver2 | 4 | ntrocas | 12 |
| nanaliza | 100 | mtrosi | 1 |
| nduramin | 9 | mtrono | 1 |
| nduravar | 9 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 1 | kseed | 3667 |
| mgarv | 3 | nmax | 4 |
| mgera | 1 | nactmax2 | 1 |
| mcarg | 1 | nplmax | 50000 |
| malea | 6 | ncl | 10 |
| nsaen | 7 | | |

**c)** parâmetros.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 0 | |

**b)** configuração ótima;

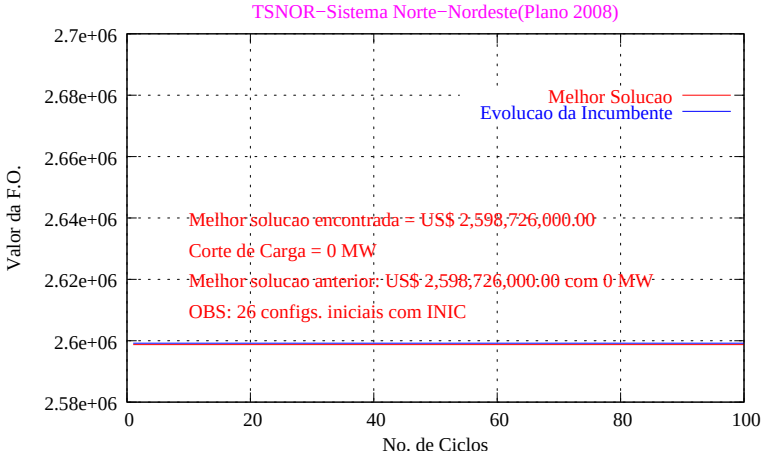
## 8.3 Sistema Norte-Nordeste 2008

Este sistema teste Norte-Nordeste brasileiro é constituído por 87 barras e 179 ramos, e para as simulações aqui consideradas foram utilizados os dados apresentados no Apêndice B, para o Plano 2008.

### 8.3.1 Algoritmo TSNOR

Na Tabela 21, está a melhor configuração encontrada com o algoritmo TSNOR e os parâmetros utilizados. Para este teste foi usado um conjunto de vinte e seis configurações iniciais obtidas com o algoritmo inicializador INIC. A melhor configuração, US$ 2,598,126,000.00 sem corte de carga, foi encontrada com um tempo de processamento de 24,0 min, foram calculados 6.947 PPLs, obtidos no 2º ciclo.

Tabela 21: Sistema Norte_Nordeste (Plano 2008) - algoritmo TSNOR.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | nsaen | 15 |
| ncorte | 150 | ncosto | 1 |
| nintens | 6 | nsem | 26 |
| ndiver1 | 3 | nsem2 | 50 |
| ndiver2 | 5 | melhor | 2 |
| nanaliza | 150 | ntrocas | 12 |
| nduramin | 10 | mtrosi | 1 |
| nduravar | 18 | mtrono | 1 |
| msens | 1 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 4 | nmax | 4 |
| mgera | 2 | nactmax2 | 1 |
| mcarg | 2 | nplmax | 50000 |
| malea | 0 | ncl | 20 |

**c)** parâmetros.

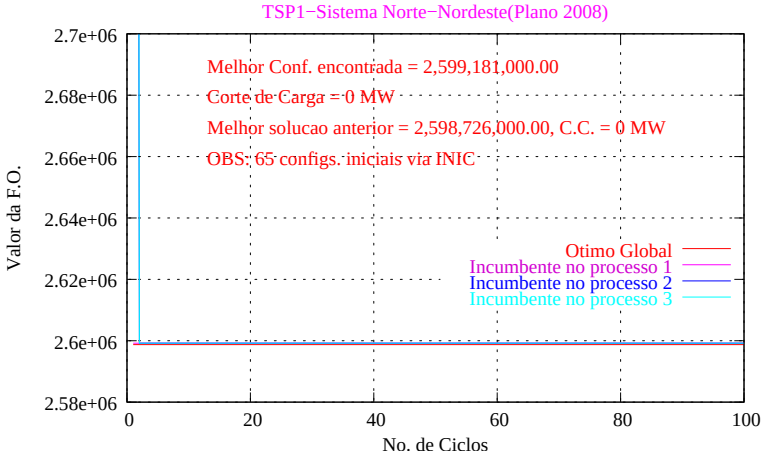
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 4 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 |
| 2 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 2 | 4 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

### 8.3.2 Algoritmo TSP1

Na Tabela 22 mostra-se a melhor configuração encontrada para o sistema Norte-Nordeste brasileiro, plano 2008, com o algoritmo TSP1, com um tempo computacional de 0,46 min, onde foram calculados 23 PPLs no 1º ciclo. Nesta simulação foram empregadas sessenta e cinco configurações iniciais obtidas com o inicilizador INIC, e obteve-se a solução de US$ 2,599,181,000.00 sem corte de carga.

Tabela 22: Sistema Norte_Nordeste (Plano 2008) - algoritmo TSP1.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 1 |
| ncorte | 100 | nsem | 65 |
| nintens | 1 | nsem2 | 50 |
| ndiver1 | 1 | melhor | 2 |
| ndiver2 | 1 | ntrocas | 12 |
| nanaliza | 200 | mtrosi | 1 |
| nduramin | 18 | mtrono | 1 |
| nduravar | 18 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 4 | nmax | 12 |
| mgera | 1 | nactmax2 | 2 |
| mcarg | 3 | nplmax | 100000 |
| malea | 12 | ncl | 20 |
| nsaen | 15 | | |

**c)** parâmetros.

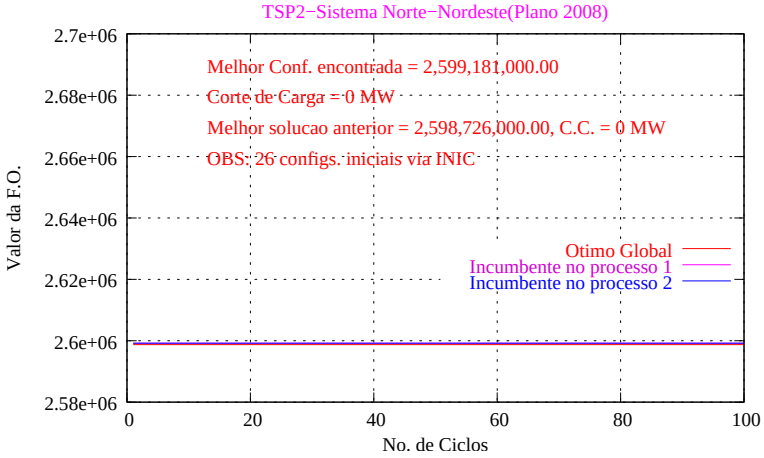
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 |
| 2 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 2 | 4 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

### 8.3.3 Algoritmo TSP2

Na Tabela 23 mostra a melhor configuração encontrada para o sistema Norte-Nordeste brasileiro, plano 2008, com o algoritmo TSP2, com um tempo computacional de 0,32 min, onde foram calculados 17 PPLs no 1º ciclo. Nesta análise foram empregadas vinte e seis configurações iniciais obtidas com o inicializador INIC, e obteve-se a solução de US$ 2,599,181,000.00 sem corte de carga.

Tabela 23: Sistema Norte_Nordeste (Plano 2008) - algoritmo TSP2.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 1 |
| ncorte | 100 | nsem | 26 |
| nintens | 2 | nsem2 | 50 |
| ndiver1 | 2 | melhor | 2 |
| ndiver2 | 5 | ntrocas | 12 |
| nanaliza | 200 | mtrosi | 1 |
| nduramin | 18 | mtrono | 1 |
| nduravar | 18 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 2 | nmax | 12 |
| mgera | 0 | nactmax2 | 2 |
| mcarg | 2 | nplmax | 100000 |
| malea | 14 | ncl | 20 |
| nsaen | 15 | | |

**c)** parâmetros.

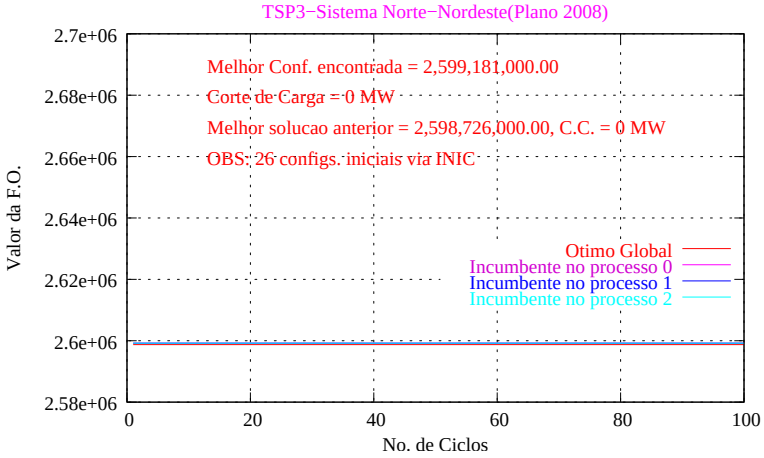
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 |
| 2 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 2 | 4 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

### 8.3.4 Algoritmo TSP3

Na Tabela 24 mostra-se a melhor configuração encontrada (US$ 2,599,181,000.00 sem corte de carga) para o sistema Norte-Nordeste brasileiro, plano 2008, com o algoritmo TSP3, com um tempo computacional de 0,59 min, onde foram calculados 30 PPLs no 1º ciclo. Nesta análise também foram empregadas vinte e seis configurações iniciais, geradas via algoritmo de inicialização INIC.

Tabela 24: Sistema Norte_Nordeste (Plano 2008) - algoritmo TSP3.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 1 |
| ncorte | 150 | nsem | 26 |
| nintens | 1 | nsem2 | 50 |
| ndiver1 | 2 | melhor | 2 |
| ndiver2 | 5 | ntrocas | 12 |
| nanaliza | 200 | mtrosi | 1 |
| nduramin | 18 | mtrono | 1 |
| nduravar | 18 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 7 | nmax | 12 |
| mgera | 1 | nactmax2 | 2 |
| mcarg | 2 | nplmax | 100000 |
| malea | 14 | ncl | 20 |
| nsaen | 15 | | |

**c)** parâmetros;

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 1 | 4 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 4 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 |
| 2 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 2 | 4 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 2 | 2 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

## 8.4 Sistema Norte-Nordeste 2002

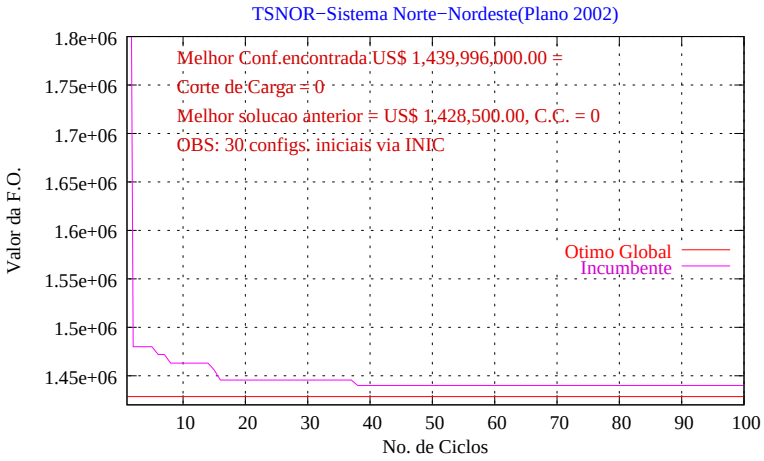
Este sistema Norte-Nordeste brasileiro teste é constituído por 87 barras e 179 ramos, e para as simulações aqui consideradas foram utilizados os dados apresentados no Apêndice B, para o Plano 2002.

A melhor solução encontrada anteriormente para esse sistema no trabalho de (DEOLIVEIRA, 2004), com o uso de algoritmos genéticos paralelos, foi de US$ 1,428,500,000.00 sem corte de carga. Na falta de um resultado específico para BT paralelo será utilizado este valor como referência.

### 8.4.1 Algoritmo TSNOR

A melhor configuração foi encontrada com um tempo de processamento de 6,0 min, foram calculados 12.896 PPLs obtido no 38º ciclo. Nesta simulação foram empregadas trinta configurações iniciais obtidas com o inicializador INIC. Os dados desta simulação são apresentados na Tabela 25, onde obteve a solução de US$ 1,439,996,000.00 sem corte de carga.

Tabela 25: Sistema Norte_Nordeste (Plano 2002) - algoritmo TSNOR.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | nsaen | 15 |
| ncorte | 50 | ncosto | 1 |
| nintens | 2 | nsem | 30 |
| ndiver1 | 2 | nsem2 | 70 |
| ndiver2 | 5 | melhor | 2 |
| nanaliza | 200 | ntrocas | 8 |
| nduramin | 18 | mtrosi | 1 |
| nduravar | 18 | mtrono | 1 |
| msens | 4 | md1 | 230000 |
| mmin1 | 2 | md2 | 45000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 3 | nmax | 12 |
| mgera | 3 | nactmax2 | 2 |
| mcarg | 0 | nplmax | 20000 |
| malea | 14 | ncl | 20 |

**c)** parâmetros.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 1 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| 2 | 1 | 1 | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 2 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

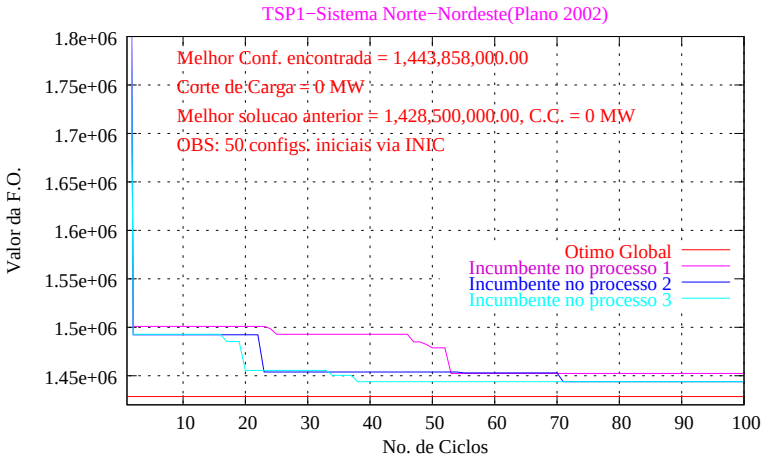
**b)** Melhor Configuração Encontrada;

### 8.4.2 Algoritmo TSP1

Na Tabela 26 mostra-se a melhor configuração encontrada para o sistema Norte-Nordeste, plano 2002, com o algoritmo TSP1, com um tempo computacional de 3,0 min, onde foram calculados 11.324 PPLs no 71º ciclo obteve-se a solução de US$ 1,443,858,000.00 sem corte de carga. Nesta simulação foram empregadas cinqüenta configurações iniciais

obtidas com o inicializador INIC.

Tabela 26: Sistema Norte_Nordeste (Plano 2002) - algoritmo TSP1.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 1 |
| ncorte | 100 | nsem | 50 |
| nintens | 2 | nsem2 | 70 |
| ndiver1 | 2 | melhor | 6 |
| ndiver2 | 5 | ntrocas | 8 |
| nanaliza | 200 | mtrosi | 1 |
| nduramin | 20 | mtrono | 1 |
| nduravar | 20 | md1 | 3000000 |
| mmin1 | 2 | md2 | 1000000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 3 | nmax | 12 |
| mgera | 2 | nactmax2 | 2 |
| mcarg | 2 | nplmax | 100000 |
| malea | 14 | ncl | 20 |
| nsaen | 15 | | |

**c)** parâmetros.

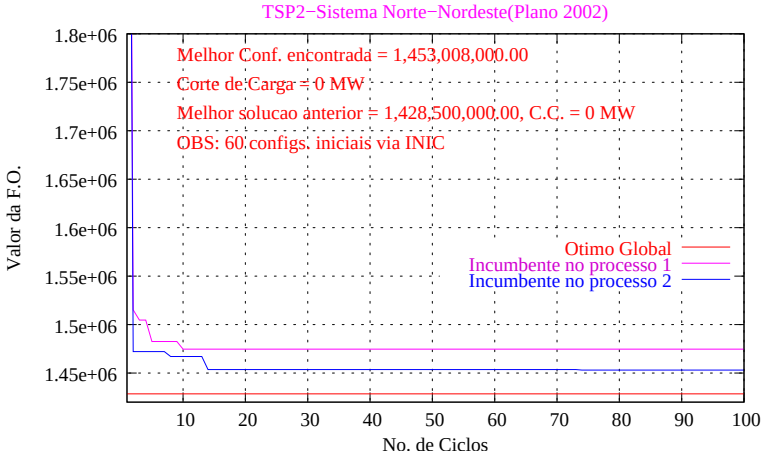
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 1 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| 2 | 1 | 1 | 0 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 2 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

### 8.4.3 Algoritmo TSP2

Na Tabela 27 mostra-se a melhor configuração encontrada para o sistema Norte-Nordeste, plano 2002, com o algoritmo TSP2 de US$ 1,453,008,000.00 sem corte de carga, com um tempo computacional de 8,0 min onde foram calculados 15.860 PPLs obtido no 74º ciclo. Nesta análise foram empregadas sessenta configurações iniciais obtidas com o inicializador INIC.

Tabela 27: Sistema Norte_Nordeste (Plano 2002) - algoritmo TSP2.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 1 |
| ncorte | 100 | nsem | 60 |
| nintens | 2 | nsem2 | 70 |
| ndiver1 | 2 | melhor | 6 |
| ndiver2 | 5 | ntrocas | 8 |
| nanaliza | 200 | mtrosi | 1 |
| nduramin | 18 | mtrono | 1 |
| nduravar | 18 | md1 | 3000000 |
| mmin1 | 2 | md2 | 1000000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 2 | nmax | 12 |
| mgera | 2 | nactmax2 | 2 |
| mcarg | 2 | nplmax | 100000 |
| malea | 14 | ncl | 20 |
| nsaen | 15 | | |

**c)** parâmetros.

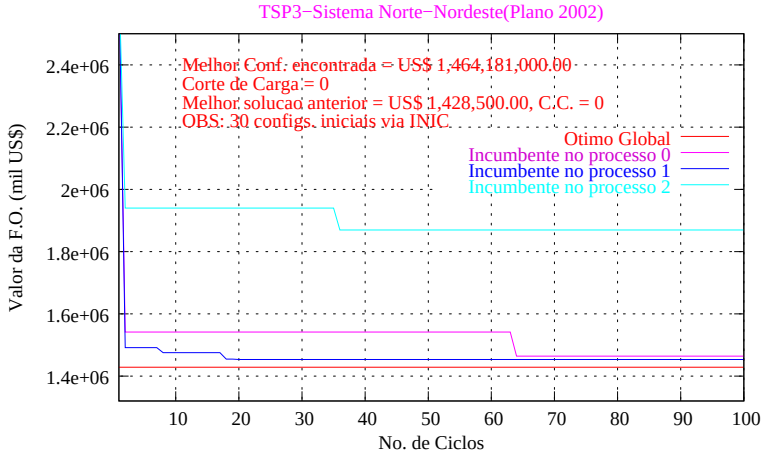
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 |
| 0 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 6 | 0 | 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

### 8.4.4 Algoritmo TSP3

Na Tabela 28 mostra-se a melhor configuração encontrada para o sistema Norte-Nordeste brasileiro, plano 2002, com o algoritmo TSP3 com valor de US$ 1,464,181,000.00 sem corte de carga com um tempo computacional de 4,17 min onde foram calculados 14.742 PPLs no 64º ciclo. Nesta análisa foram empregadas trinta configurações iniciais geradas via algoritmo de inicialização INIC.

Tabela 28: Sistema Norte_Nordeste (Plano 2002) - algoritmo TSP3.




**a)** gráfico;

| parâmetro | valor | parâmetro | valor |
|---|---|---|---|
| nafa | 1000 | ncosto | 1 |
| ncorte | 150 | nsem | 30 |
| nintens | 2 | nsem2 | 60 |
| ndiver1 | 2 | melhor | 6 |
| ndiver2 | 5 | ntrocas | 8 |
| nanaliza | 200 | mtrosi | 1 |
| nduramin | 18 | mtrono | 1 |
| nduravar | 18 | md1 | 3000000 |
| mmin1 | 2 | md2 | 1000000 |
| mmin2 | 2 | kseed | 3667 |
| mgarv | 3 | nmax | 12 |
| mgera | 3 | nactmax2 | 2 |
| mcarg | 0 | nplmax | 100000 |
| malea | 14 | ncl | 20 |
| nsaen | 15 | | |

**c)** parâmetros.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 |
| 0 | 0 | 3 | 0 | 2 | 0 | 0 | 0 | 6 | 0 | 3 | 0 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 2 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

**b)** Melhor Configuração Encontrada;

## 8.5 Resumo dos Resultados

Nesta seção, são apresentados os resultados de diversas simulações realizadas com as diferentes versões implementadas neste trabalho para a resolução do problema do planejamento da expansão da transmissão, para os exemplos de redes de pequeno, médio e grande porte citados anteriormente.

Na Tabela 29 a seguir, são apresentados os resultados de simulações, para os diferentes sistemas testes, com o algoritmo serial TSNOR.

Tabela 29: Resultados gerais com algoritmo Busca Tabu serial - TSNOR.

| algoritmo / itens / sistemas | TSNOR | | | |
|---|---|---|---|---|
| | Número de Ciclos da Solução ótima | Número de PPLs da Solução òtima | Tempo da Solução ótima (s) | Tempo Médio por PPL |
| Garver c/ redesp. | 2 | 306 | 6,4 | 0,0209 |
| Garver s/ redesp. | 3 | 194 | 4,1 | 0,0211 |
| Sul Bras. c/ redesp. | 13 | 1.461 | 46,1 | 0,0316 |
| Sul Bras. s/ redesp. | 12 | 1.868 | 63,2 | 0,0338 |
| Nor-Nord.-2008 | 2 | 6.947 | 24 min [a] | 0,2073 |
| Nor-Nord.-2002 | 38 | 12.896 | 6 min [b] | 0,0279 |

[a]ótimo local US$ 2,598,126,000.00 - corte de carga 0 MW
[b]ótimo local US$ 1,439,996,000.00 - corte de carga 0 MW

Na Tabela 30 a seguir, são apresentados os resultados de simulações, para os diferentes sistemas testes, com o algoritmo TSP1.

Tabela 30: Resultados gerais com algoritmo Busca Tabu paralelo - TSP1

| algoritmo / itens / sistemas | TSP1 | | | |
|---|---|---|---|---|
| | Número de Ciclos da Solução ótima | Número de PPLs da Solução òtima | Tempo da Solução ótima (s) | Tempo Médio por PPL |
| Garver c/ redesp. | 2 | 224 | 0,44 | 0,0020 |
| Garver s/ redesp. | 2 | 51 | 0,11 | 0,0022 |
| Sul Bras. c/ redesp . | 4 | 873 | 6,3 | 0,0072 |
| Sul Bras. s/ redesp. | 22 | 1.955 | 15,0 | 0,0077 |
| Nor-Nord.-2008 | 1 | 23 | 0,46 [a] | 0,0200 |
| Nor-Nord.-2002 | 71 | 11.324 | 3 min [b] | 0,0159 |

[a]ótimo local US$ 2,599,181,000.00 - corte de carga 0 MW
[b]ótimo local US$ 1,453,858,000.00 - corte de carga 0 MW

Na Tabela 31 a seguir, são apresentados os resultados de simulações, para os diferentes sistemas testes, com o algoritmo TSP2.

Tabela 31: Resultados gerais com algoritmo Busca Tabu paralelo - TSP2

| algoritmo / itens / sistemas | TSP2 | | | |
|---|---|---|---|---|
| | Número de Ciclos da Solução ótima | Número de PPLs da Solução òtima | Tempo da Solução ótima (s) | Tempo Médio por PPL |
| Garver c/ redesp. | 2 | 143 | 0,47 | 0,0033 |
| Garver s/ redesp. | 2 | 135 | 0,31 | 0,0023 |
| Sul Bras. c/ redesp . | 2 | 966 | 7,7 | 0,0080 |
| Sul Bras. s/ redesp. | 4 | 581 | 4,4 | 0,0076 |
| Nor-Nord.-2008 | 1 | 17 | 0,32 [a] | 0,0188 |
| Nor-Nord.-2002 | 74 | 15.860 | 8 min [b] | 0,0303 |

[a]ótimo local US$ 2,599,181,000.00 - corte de carga 0 MW
[b]ótimo local US$ 1,453,008,000.00 - corte de carga 0 MW

Na Tabela 32 a seguir, são apresentados os resultados de simulações, para os diferentes sistemas testes, com o algoritmo TSP3.

Tabela 32: Resultados gerais com algoritmo Busca Tabu paralelo - TSP3

| algoritmo / itens / sistemas | TSP3 | | | |
|---|---|---|---|---|
| | Número de Ciclos da Solução ótima | Número de PPLs da Solução òtima | Tempo da Solução ótima (s) | Tempo Médio por PPL |
| Garver c/ redesp. | 2 | 641 | 1,0 | 0,0016 |
| Garver s/ redesp. | 2 | 255 | 0,71 | 0,0028 |
| Sul Bras. c/ redesp . | 2 | 2.305 | 16,3 | 0,0071 |
| Sul Bras. s/ redesp. | 5 | 794 | 5,9 | 0,0074 |
| Nor-Nord.-2008 | 1 | 30 | 0,59 [a] | 0,0197 |
| Nor-Nord.-2002 | 64 | 14.742 | 4,17 min [b] | 0,0170 |

[a]ótimo local US$ 2,599,181,000.00 - corte de carga 0 MW
[b]ótimo local US$ 1,464,181,000.00 - corte de carga 0 MW

Nas Tabelas 33, 34, 35, 36, 37 e 38, a seguir, são apresentados os resultados do *speedup* e da eficiência de todos os sistemas testes simulados, para as versões paralelas TSP1, TSP2 e TSP3. A base da comparação foi o tempo de processamento dos diferentes algoritmos paralelos em relação ao tempo de processamento do algoritmo serial e também levando

em conta o número de processadores utilizados.

Tabela 33: Speedup e eficiência dos algoritmos (Garver c/redespacho).

| algoritmo | tempo (s) | nº de procs | *speedup* | eficiência (%) |
|---|---|---|---|---|
| TSNOR | 6,4 | - | - | - |
| TSP1 | 0,44 | 3 | 14,5 | 485 |
| TSP2 | 0,47 | 2 | 13,6 | 681 |
| TSP3 | 1,0 | 3 | 6,4 | 213 |

Tabela 34: Speedup e eficiência dos algoritmos (Garver s/redespacho).

| algoritmo | tempo (s) | nº de procs | *speedup* | eficiência (%) |
|---|---|---|---|---|
| TSNOR | 4,1 | - | - | - |
| TSP1 | 0,11 | 3 | 37,3 | 1.245 |
| TSP2 | 0,31 | 2 | 13,2 | 661 |
| TSP3 | 0,71 | 3 | 5,8 | 193 |

Tabela 35: Speedup e eficiência dos algoritmos (Sul c/redespacho).

| algoritmo | tempo (s) | nº de procs | *speedup* | eficiência (%) |
|---|---|---|---|---|
| TSNOR | 46,1 | - | - | - |
| TSP1 | 6,3 | 3 | 7,3 | 244 |
| TSP2 | 7,7 | 2 | 6,0 | 300 |
| TSP3 | 16,3 | 3 | 2,8 | 93 |

Tabela 36: Speedup e eficiência dos algoritmos (Sul s/redespacho).

| algoritmo | tempo (s) | nº de procs | *speedup* | eficiência (%) |
|---|---|---|---|---|
| TSNOR | 63,2 | - | - | - |
| TSP1 | 15,0 | 3 | 4,2 | 140 |
| TSP2 | 4,4 | 2 | 14,4 | 720 |
| TSP3 | 5,9 | 3 | 10,7 | 357 |

Tabela 37: Speedup e eficiência dos algoritmos (Nor2008).

| algoritmo | tempo (min) | nº de procs | *speedup* | eficiência (%) |
|---|---|---|---|---|
| TSNOR | 24 | - | - | - |
| TSP1 | 0,46 | 3 | 52,2 | 1.739 |
| TSP2 | 0,32 | 2 | 75 | 3.750 |
| TSP3 | 0,59 | 3 | 40,7 | 1.356 |

Tabela 38: Speedup e eficiência dos algoritmos (Nor2002).

| algoritmo | tempo (min) | nº de procs | *speedup* | eficiência (%) |
|---|---|---|---|---|
| TSNOR | 6,0 | - | - | - |
| TSP1 | 3,0 | 3 | 2,0 | 66,7 |
| TSP2 | 8,0 | 2 | 0,75 | 37.5 |
| TSP3 | 4,17 | 3 | 1,44 | 48 |

Observa-se pelos resultados que na maioria dos casos analisados, para os sistemas Garver e Sul brasileiro, as versões paralelas apresentam além de um *speedup* alto uma eficácia elevada. Por exemplo, para o algoritmo TSP1 simulado para o sistema Garver sem redespacho Tabela 34, tem-se que o mesmo foi aproximadamente doze vezes mais eficiente que o TSNOR para encontrar a solução ótima. Para este mesmo sistema teste os algoritmos TSP2 e TSP3 foram aproximadamente sete e duas vezes mais eficientes que o TSNOR, respectivamente.

Comparando os resultados obtidos com as versões paralelas para o sistema Norte-Nordeste brasileiro plano 2008, em relação ao algoritmo TSNOR, pode-se ver que o *speedup* e a eficiência foram extremamente altos. Valores de dezessete, trinta e sete e treze vezes mais em eficiência foram encontrados para os algoritmos TSP1, TSP2 e TSP3, respectivamente. Observou-se que dentro das configurações iniciais já existia a melhor configuração encontrada pelos algoritmos, o que não inviabiliza a análise pois também no algoritmo serial foi usada esta mesma configuração.

Por outro lado, para o sistema Norte-Nordeste brasileiro plano 2002, os algoritmos TSP1, TSP2 e TSP3 foram aproximadamente uma e meia, três e duas vezes menos eficientes que o TSNOR, respectivamente. Estes últimos resultados da Tabela 38, ocorreram principalmente pela dificuldade na calibração dos parâmetros.

# 9 CONCLUSÕES E SUGESTÕES PARA FUTUROS TRABALHOS

Neste trabalho foram apresentados diferentes algoritmos para a resolução do problema do planejamento da expansão da transmissão de energia elétrica, tratando do ponto de vista estático, envolvendo a metaheurística Busca Tabu.

A Busca Tabu é uma técnica que surgiu de conceitos da inteligência artificial e é um procedimento de busca que explora o espaço de soluções além do ótimo local. A BT permite passar para uma solução da vizinhança de cada configuração corrente mesmo que não seja tão boa como a corrente, escapando de ótimos locais e continuando a busca para soluções melhores.

Com base nos resultados apresentados no capítulo anterior conclui-se que os algoritmos BT paralelos desenvolvidos apresentaram em sua totalidade tempos de processamento menores que os tempos obtidos com o algoritmo BT serial.

Para sistemas de pequeno e médio portes estes tempos foram bem menores que os esperados, resultando em valores de *speedup* e eficiência elevados. Isso se deve a forma da calibração dos parâmetros que é feita de maneira diferente para as versões paralelas e serial. Cabe lembrar que os resultados apresentados são os melhores encontrados para cada algoritmo, dentro de uma grande quantidade de simulações efetuadas.

Para sistemas de grande porte, este comportamento se comprovou para o sistema Norte-Nordeste plano 2008, mas para o sistema Norte-Nordeste plano 2002 os resultados apresentados indicam valores de *speedup* e eficiência abaixo dos esperados. Neste caso em particular foram feitas poucas simulações e não se pode fazer ainda uma análise conclusiva.

De uma maneira geral, a técnica de BT mostrou-se eficiente em encontrar valores ótimos para sistemas de pequeno e médio porte, e boas soluções para sistemas de grande

porte. Mas durante o desenvolvimento deste trabalho constatou-se que pela grande variedade de alternativas de estratégias de busca possíveis de serem implementadas no algoritmo, outras variantes dos algoritmos propostos poderiam ter sido analisadas.

Assim, um exemplo de um trabalho futuro a ser desenvolvido seria a implementação de um algoritmo paralelo em que cada processador trabalha-se especificamente com uma técnica de BT diferente, ou seja, enquanto um processador trabalha com a estratégia de diversificação o outro trabalha com a estratégia de intensificação, bem como em outros processadores poderiam trabalhar usando outras estratégias.

Outra proposta de trabalho futuro seria implementar algoritmos de BT que tenham na sua inicialização o uso de algoritmos heurísticos construtivos específicos, ou seja, cada processador trabalharia com algoritmos BT diferentes em sua inicialização. Esta proposta, entretanto, já está contemplada naqueles algoritmos paralelos propostos em que se pode calibrar separadamente os arquivos de parâmetros.

# *Referências*

AL-TAWIL, K. E.; MORITZ, C. A. Performance modeling and evaluation of MPI. *Journal of Parallel and Distributed Computing*, New York, v. 61, p. 202-223, 2001.

ALASDAIRA, R.; BRUCE, A.; MILLS, J. G.; SMITH, A. G. *CHIMP:* version 2.0: user guide, 1994.

BARNEY, B. M. *Message passing interface (MPI).* [s.l.]: LNLL, 2002. Disponível em: <`http://www.llnl.gov/computing/tutorials/workshops/workshop/home/MAIN.html`>. Acesso em: 02 out. 2006.

BARNEY, B. M. *Introduction to parallel computing.* [s.l.]: LNLL, 2006. Disponível em: <`http://www.llnl.gov/computing/tutorials/parallel_comp/`>. Acesso em: 27 set. 2006.

BURNS, G.; DAOUD, R.; VAIGL, J. *LAM:* an open cluster environment for MPI. Ohio: Ohio Supercomputer Center Columbus, [s.n.], 1994.

DE OLIVEIRA, S. A. *Metaheurísticas aplicadas ao planejamento da expansão da transmissão de energia elétrica em ambiente de processamento distribuído.* 2004. 182 f. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2004.

DE OLIVEIRA, S. A.; DE ALMEIDA, C. R. T.; MONTICELLI, A. Times assíncronos aplicados a métodos heurísticos construtivos de planejamento da expansão da transmissão. In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 12., 1998, Uberlândia. *Anais...* Uberlândia: SBA-UFU, v. 3, p. 1029-1034, 1998.

DE OLIVEIRA, S. A.; DE ALMEIDA, C. R. T.; MONTICELLI, A. Times assíncronos inicializador de métodos combinatoriais para o planejamento da expansão da transmissão. SEMINÁRIO NACIONAL DE PRODUÇÃO E TRANSMISSÃO DE ENERGIA ELÉTRICA - SNPTEE, 15., 1999, Foz do Iguaçu. *Anais...* Foz do Iguaçu: Cigré Brasil/Itaipu Binacional, 1999. Grupo VII-GPL/IT 04.

DE OLIVEIRA, S. A.; GALLEGO, R. A.; ESCOBAR, A.; RESTREPO, R. Parallel combinatorial algorithm for statical planning of a transmission system. In: INTERNATIONAL CONFERENCE ON CAD/CAM, ROBOTICS AND FACTORIES AT THE FUTURE, 2001, Durban. *Anais...* Durban: v. 17, p. 1183-1191, 2001.

GALLEGO, A. R.; ROMERO, R.; MONTICELLI, J. A. Transmission system expansion planning by simulated annealing. *IEEE Transactions on Power Systems*, New York, v. 11, n. 1, 1996.

GALLEGO, A. R.; ROMERO, R.; MONTICELLI, J. A. Tabu search algorithm for network synthesis. *IEEE Transactions on Power Systems*, New York, v. 15, n. 2, p. 490-495, 2000.

GALLEGO, R. A. *Planejamento a longo prazo de sistemas de transmissão usando técnicas de otimização combinatorial*. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 1997.

GALLEGO, R. A.; ALVES, A. B.; MONTICELLI, A.; ROMERO, R. Parallel simulated annealing applied to long term transmission network expansion planning. *IEEE Transactions on Power Systems*, New York, v. 12, n. 1, p. 181–188, 1997.

GALLEGO, R. A.; DE OLIVEIRA, S. A. *TSNOR:* modelo de "tabu search" para o planejamento da transmissão de sistemas de grande porte. Campinas: FEEC/Unicamp, março 1997. 6 p.

GARVER, L. Transmission network estimation using linear programming. *IEEE Transactions on Power Apparatus and Systems.*, New York, PAS-89, n. 7, p. 1688–1697, 1970.

GLOVER, F. *Tabu search fundamental and uses.* [s.l.]: University of Colorado, Graduate School of Business, 1986.

GLOVER, F.; LAGUNA, M. *Tabu search in modern heuristic techniques for combinatorial problems*. Oxford: Blackwell Scientific, 1993.

GLOVER, F.; LAGUNA, M. *Tabu search*. Massachusetts: Klüwer Academic Publishers, 1997.

GOLDBERG, D. E. *Genetic algorithms in search, optimization and machine learning.* Reading: Addison Wesley, 1989.

GROPP, B.; LUSK, R.; SKJELLUM, T. Portable MPI model implementation. Argonne: Argonne National Laboratory, 1994.

HPFF. *High performance FORTRAN:* conceitos e definições, 2006. Disponível em: <`http://www.crpc.rice.edu/HPFF`>. Acesso em: 28 set. 2006.

LAGUNA, M. *Tabu search tutorial*. [S.l.]: I ESCUELA DE VERANO LATINO-AMERICANA DE INVESTIGACION OPERATIVA, 1995. 112 p.

LATORRE, G. *Static models for long-term transmission planning*. Thesis (Doctoral Dissertation) — Universidad Pontificia Comillas, Madrid, Spain, 1993.

MONTICELLI, A.; SANTOS JR., A.; PEREIRA, M. V. F.; CUNHA, S. H.; PRAÇA, J. C. G.; PARK, B. J. Interactive transmission network planning using a lest-effort criterion. *IEEE Transactions on Power Apparatus and Systems*, New York, v. 101, n. 10, p. 3919-3925, 1982.

MONTICELLI, A. J.; SHIOZER, D. J.; SOUZA, H. *Uma introdução ao PVM:* como a paralelização pode ajudar a resolver problemas complexos de otimização. v. 18, p. 63-73, 1998. Disponível em: <`http://www.dep.fem.unicamp.br/unisim/english/publications.html`>. Acesso em: 26 set. 2006.

MOURA; SILVA, L.; BUYYA, R. Parallel programming models and paradigms. In: BUYYA, R. (Org.) *High performance cluster computing:* programming and applications. New Jersey: Prentice Hall, PTR, 1999, p. 4-27.

MPIFORUM. *MPI:* a message passing interface standard. Knoxville: University of Tennessee, 1995. Disponível em: <`http://www.mpi-forum.org/`>. Acesso em: 01 fev. 2008.

NUPAIROJ, N.; NI, L. M. Performance evaluation of some MPI implementations on workstation clusters. In: SCALABLE PARALLEL LIBRARIES CONFERENCE, 1994, [s.l.] *Proceedings...* [s.l.]: IEEE Computer Society Press, 1994. p. 98-105.

PEREIRA, M.; PINTO, L. Aplication of sensitivity analysis of load supplying capability to interactive transmission expansion planning. *IEEE Transactions on Power Apparatus and Systems*, New York, v. PAS-104, n. 2, p. 381-389, 1985.

PÉREZ-ARRIAGA, J. I.; GÓMEZ, T.; RAMOS, A. *State-of-the-art status on transmission networks planning.* Madri: Instituto de Investigación Tecnológica, 1987.

PRAMANICK, I. Parallel programming languages and environments. In: BUYYA, R. (Org.) *High performance cluster computing:* programming and applications. New Jersey: Prentice Hall, PTR, 1999. p. 28-47.

PUNTEL, W. R.; MERRILL, H. M.; SAGER, M. A.; WOOD, J. A. *Power system planning techniques course.* [s.l.]: Power Technologies, 1984.

ROCHA, C. R. M. *Desenvolvimento de técnicas heurísticas e de otimização clássica para o problema de planejamento da expansão a longo prazo de sistemas de transmissão*, 2004. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, 2004.

ROCHA, C. R. M.; MANTOVANI, M.; ROMERO, R.; MANTOVANI, J. R. S. Evolution of hybrid models for static and multistage transmission system planning. *IEE Proceedings - Generation, Transmission and Distribution*, London, v. 150, n. 5, 2003.

ROMERO, R.; HAFFNER, S.; MONTICELLI, A.; GARCIA, A. Specialised branch-and-bound algorithm for transmission network expansion planning. *IEE Proceedings - Generation, Transmission and Distribution*, London, v. 148, n. 5, 2001.

ROMERO, R.; MONTICELLI, A.; GARCIA, A.; HAFFNER, S. Test systems and mathematical models for transmission network expansion planning. *IEE Proceedings - Generation, Transmission and Distribution*, London, v. 149, n. 1, 2002.

ROMERO, R.; MANTOVANI, J. R. S. Introdução as metaheurísticas. In: CONGRESSO TEMÁTICO DE DINÂMICA E CONTROLE, 3., 2004, Ilha Solteira. *Anais...* Ilha Solteira: SBMAC, 2004.

ROMERO, R.; ROCHA, C.; MANTOVANI, M.; MANTOVANI, J. R. S. Evalution of hybrid models for static and multistage transmission systems planning. *Submetido a Revista Internacional*, v. 18, n. 1, janeiro, fevereiro, março 2007.

SILVA, E. L.; AREIZA, J. M.; GIL, H. A. Planejamento da expansão baseados em métodos heurísticos. In: SIMPÓSIO DE ESPECIALISTAS EM PLANEJAMENTO DA OPERAÇÃO E EXPANSÃO ELÉTRICA, 7., Curitiba. *Anais...* Curitiba: UFPR/Copel, 2000.

SILVA, E. L.; AREIZA, J. M.; OLIVEIRA, G. C.; BINATO, S. Transmission network expansion planning under a tabu search approach. *IEEE Transactions on Power Systems*, New York, v. 16, n. 1, p. 62-68, 2001.

SULLIVAN, R. L. *Power system planning*. New York: McGraw-Hill, 1977.

VILLASANA, R.; GARVER, L.; SALON, S. Transmission network planning using linear programming. *IEEE Transactions on Power Systems*, New York, PAS-104, n. 2, p. 349–356, 1985.

WEN, F.; CHANG, C. S. Transmission network optimal planning using the tabu search method. *Electric Power Systems Research*, Lausanne, n. 42, p. 153–163, 1997.

# *APÊNDICE A – Uso do Ambiente LAM/MPI*

Abaixo, descreve-se os passos para preparar uma sessão MPI utilizando LAM/MPI.

## A.1 Preparando o ambiente MPI

As variáveis de ambiente precisam ser adaptadas de acordo com a configuração do seu sistema.

## A.2 O arquivo Hostfile

Em seu diretório de trabalho (onde seus arquivos binários MPI vão ficar), crie um arquivo *hostfile* que fornece a lista de máquinas a serem incluídas em uma sessão MPI. A seguir encontra-se um exemplo de um arquivo *"hostfile"*:

fiona.dee.feis.unesp.br
robim.dee.feis.unesp.br
pigeotto.dee.feis.unesp.br

IMPORTANTE: Assegure-se que para cada máquina listada no seu arquivo *"hostfile"* existe uma entrada correspondente no seu arquivo *.shosts* (arquivo com todas as máquinas que têm o MPI instalado). Por exemplo, o seu arquivo *.shosts* pode conter:

fiona.dee.feis.unesp.br
robim.dee.feis.unesp.br
pigeotto.dee.feis.unesp.br
coyote.dee.feis.unesp.br
aramis.dee.feis.unesp.br

## A.3 Começando com o LAM

### A.3.1 Lamboot

Toda sessão MPI deve ser inicializada com um e somente um comando *lamboot*. Para iniciar uma sessão MPI sob o LAM, no prompt do Linux, o comando usado para a montagem das máquinas é:

*lamboot -v hostfile*

A opção -v é para habilitar o modo *verbose*, de modo que se possa observar o que o LAM está fazendo e o *hostfile* é o arquivo onde estão as máquinas, no diretório de trabalho. Se o comando funcionar corretamente, deve-se ver algo assim (dependendo do seu *hostfile*):

LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University

$n-1<13956>$ ssi:boot:base:linear: booting n0 (pigeotto)
$n-1<13956>$ ssi:boot:base:linear: booting n1 (robim)
$n-1<13956>$ ssi:boot:base:linear: booting n2 (fiona)
$n-1<13956>$ ssi:boot:base:linear: finished

A maneira mais simples de compilar os programas MPI sendo executadas sob a implementação do LAM é modificar um *Makefile* existente. Modifique esse *Makefile* para o seu ambiente e insira novos comandos à medida que se sentir mais confortável com o LAM.

Se tiver problemas com o comando *lamboot*, verifique o site LAM FAQ $http://www.lam-mpi.org/faq/$ na seção "*Booting* LAM"que contém várias informações.

### A.3.2 O Comando Tping N

Um modo simples de verificar o estado de cada máquina em uma sessão corrente MPI é utilizar o comando *tping N*. Tping pode ser de grande ajuda quando suspeitar-se que um programa MPI travou ou quando não se observa os resultados esperados. A opção N na linha de comando é para indicar ao LAM para testar todas as máquinas do arquivo *hostfile*. A não ser que se indique para tping o número específico de vezes para testar, o comamdo será executado até que as teclas CTRL$-c$ sejam digitadas. Pode-se usar a opção -c para especificar o número de testes a serem executados:

tping $-c3$ N.

## A.4 Compilando programas MPI

### A.4.1 SPMD

Para processar seus arquivos MPI executáveis, utilize o comando *mpirun*. Por exemplo, para executar o algoritmo busca tabu paralelo mostrado anteriormente (que se chama TSP1) usa-se:

*mpirun -np N tsp1*

A opção *N* indica que o programa deve ser executado por todas as máquinas do seu arquivo *hostfile*.

*mpirun* possui várias opções:

**-lamd**: Use o modo de envio *"daemon based"*. Isso significa que todas as mensagens irão passar através dos *daemons* LAM ao invés de ir direto de um programa MPI para outro (chamado modo *"client-to-client"*, que é o modo *default* do MPI). Os *daemons* LAM possuem capacidade para monitorar os programas MPI que estão sendo executados.

**-O**: Essa opção indica que todas as máquinas que serão utilizadas são homogêneas, não sendo necessária nenhuma conversão de dados. Essa opção não é necessária, mas pode fazer com que seu programa execute mais rápido, já que o LAM não terá que verificar se conversões devem ser feitas.

**N**: Como já mencionado, essa opção indica que o programa deve ser executado em todas as máquinas do arquivo *hostfile*. Se pretende-se executar o programa em um número menor de máquinas, pode-se especificar um subconjunto de máquinas. Por exemplo, a opção n0-n2 especifica que o programa deve ser executado nas três primeiras máquinas do arquivo *hostfile*. O comando seria:

*mpirun -np 3 tsp1*

## A.5 Lamclean

Para interromper todos os processos MPI e apagar todas as mensagens pendentes, use o comando *lamclean*:

*lamclean -v* ou apenas *lamclean*

## A.6 Finalizando o LAM

É extremamente importante que cada sessão MPI seja finalizada com o comando do LAM para finalização: *wipe*. Para finalizar de modo correto uma sessão MPI, entre com o comando:

*wipe -v hostfile*

Você deve observar que todos as máquinas especificadas no arquivo *hostfile* serão retiradas. Para o arquivo *hostfile* utilizado como exemplo, ter-se-ia:

LAM 7.1.1/MPI 2 C++/ROMIO - Indiana University

$n-1<14118>$ ssi:boot:base:linear: booting n0 (pigeotto)
$n-1<14118>$ ssi:boot:base:linear: booting n1 (robim)
$n-1<14118>$ ssi:boot:base:linear: booting n2 (fiona)
$n-1<14118>$ ssi:boot:base:linear: finished

# *APÊNDICE B – Dados dos Sistemas Testes*

- **Sistema de 06 Barras e 15 Ramos de Garver**

- **Sistema Sul Brasileiro de 46 Barras e 79 Ramos**

- **Sistema Norte-Nordeste Brasileiro de 84 Barras e 179 Ramos**

# B.1 Sistema Garver (06 Barras/15 Ramos)

Tabela 39: Dados de barras - Garver.

| Barra | Capacidade de Geração (MW) | Geração Atual (MW) | 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 |
| Total | 1.110 | 760 | 760 |

Tabela 40: Dados de linhas - Garver.

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo |
|---|---|---|---|---|---|
| 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 (46 Barras/79 Ramos)

Tabela 41: Dados de barras - Sul brasileiro

| Barra | Capacidade de Geração (MW) | Geração Atual (MW) | 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 | 1.257,0 | 944,0 | 0,0 |
| 15 | 0,0 | 0,0 | 0,0 |
| 16 | 2.000,0 | 1.366,0 | 0,0 |
| 17 | 1.050,0 | 1.000,0 | 0,0 |
| 18 | 0,0 | 0,0 | 0,0 |
| 19 | 1.670,0 | 773,0 | 0,0 |
| 20 | 0,0 | 0,0 | 1.091,2 |
| 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 |
| 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 |

Tabela 42: Dados de linhas - Sul brasileiro.

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 1 | 01-07 | 1 | 0,0616 | 270 | 4,349.00 |
| 2 | 01-02 | 2 | 0,1065 | 270 | 7,076.00 |
| 3 | 04-09 | 1 | 0,0924 | 270 | 6,217.00 |
| 4 | 05-09 | 1 | 0,1173 | 270 | 7,732.00 |
| 5 | 05-08 | 1 | 0,1132 | 270 | 7,480.00 |
| 6 | 07-08 | 1 | 0,1023 | 270 | 6,823.00 |
| 7 | 04-05 | 2 | 0,0566 | 270 | 4,046.00 |
| 8 | 02-05 | 2 | 0,0324 | 270 | 2,581.00 |
| 9 | 08-13 | 1 | 0,1348 | 240 | 8,793.00 |
| 10 | 09-14 | 2 | 0,1756 | 220 | 11,267.00 |
| 11 | 12-14 | 2 | 0,0740 | 270 | 5,106.00 |
| 12 | 14-18 | 2 | 0,1514 | 240 | 9,803.00 |
| 13 | 13-18 | 1 | 0,1805 | 220 | 11,570.00 |
| 14 | 13-20 | 1 | 0,1073 | 270 | 7,126.00 |
| 15 | 18-20 | 1 | 0,1997 | 200 | 12,732.00 |
| 16 | 19-21 | 1 | 0,0278 | 1.500 | 32,632.00 |
| 17 | 16-17 | 1 | 0,0078 | 2.000 | 10,505.00 |
| 18 | 17-19 | 1 | 0,0061 | 2.000 | 8,715.00 |
| 19 | 14-26 | 1 | 0,1614 | 220 | 10,409.00 |
| 20 | 14-22 | 1 | 0,0840 | 270 | 5,712.00 |
| 21 | 22-26 | 1 | 0,0790 | 270 | 5,409.00 |
| 22 | 20-23 | 2 | 0,0932 | 270 | 6,268.00 |
| 23 | 23-24 | 2 | 0,0774 | 270 | 5,308.00 |
| 24 | 26-27 | 2 | 0,0832 | 270 | 5,662.00 |
| 25 | 24-34 | 1 | 0,1647 | 220 | 10,611.00 |
| 26 | 24-33 | 1 | 0,1448 | 240 | 9,399.00 |
| 27 | 33-34 | 1 | 0,1265 | 270 | 8,288.00 |
| 28 | 27-36 | 1 | 0,0915 | 270 | 6,167.00 |
| 29 | 27-38 | 2 | 0,2080 | 200 | 13,237.00 |
| 30 | 36-37 | 1 | 0,1057 | 270 | 7,025.00 |
| 31 | 34-35 | 2 | 0,0491 | 270 | 3,591.00 |

continua na próxima página

Tabela 42: Dados de linhas - Sul brasileiro.
(continuação)

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 32 | 35-38 | 1 | 0,1980 | 200 | 12,631.00 |
| 33 | 37-39 | 1 | 0,0283 | 270 | 2,329.00 |
| 34 | 37-40 | 1 | 0,1281 | 270 | 8,389.00 |
| 35 | 37-42 | 1 | 0,2105 | 200 | 13,388.00 |
| 36 | 39-42 | 3 | 0,2030 | 200 | 12,934.00 |
| 37 | 40-42 | 1 | 0,0932 | 270 | 6,268.00 |
| 38 | 38-42 | 3 | 0,0907 | 270 | 6,116.00 |
| 39 | 32-43 | 1 | 0,0309 | 1.400 | 35,957.00 |
| 40 | 42-44 | 1 | 0,1206 | 270 | 7,934.00 |
| 41 | 44-45 | 1 | 0,1864 | 200 | 11,924.00 |
| 42 | 19-32 | 1 | 0,0195 | 1.800 | 23,423.00 |
| 43 | 46-19 | 1 | 0,0222 | 1.800 | 26,365.00 |
| 44 | 46-16 | 1 | 0,0203 | 1.800 | 24,319.00 |
| 45 | 18-19 | 1 | 0,0125 | 600 | 8,178.00 |
| 46 | 20-21 | 1 | 0,0125 | 600 | 8,178.00 |
| 47 | 42-43 | 1 | 0,0125 | 600 | 8,178.00 |
| 48 | 02-04 | 0 | 0,0882 | 270 | 5,965.00 |
| 49 | 14-15 | 0 | 0,0374 | 270 | 2,884.00 |
| 50 | 46-10 | 0 | 0,0081 | 2.000 | 10,889.00 |
| 51 | 04-11 | 0 | 0,2246 | 240 | 14,247.00 |
| 52 | 05-11 | 0 | 0,0915 | 270 | 6,167.00 |
| 53 | 46-06 | 0 | 0,0128 | 2.000 | 16,005.00 |
| 54 | 46-03 | 0 | 0,0203 | 1.800 | 24,319.00 |
| 55 | 16-28 | 0 | 0,0222 | 1.800 | 26,365.00 |
| 56 | 16-32 | 0 | 0,0311 | 1.400 | 36,213.00 |
| 57 | 17-32 | 0 | 0,0232 | 1.700 | 27,516.00 |
| 58 | 19-25 | 0 | 0,0325 | 1.400 | 37,748.00 |
| 59 | 21-25 | 0 | 0,0174 | 2.000 | 21,121.00 |
| 60 | 25-32 | 0 | 0,0319 | 1.400 | 37,109.00 |
| 61 | 31-32 | 0 | 0,0046 | 2.000 | 7,052.00 |

continua na próxima página

Tabela 42: Dados de linhas - Sul brasileiro.
(continuação)

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 62 | 28-31 | 0 | 0,0053 | 2.000 | 7,819.00 |
| 63 | 28-30 | 0 | 0,0058 | 2.000 | 8,331.00 |
| 64 | 27-29 | 0 | 0,0998 | 270 | 6,672.00 |
| 65 | 26-29 | 0 | 0,0541 | 270 | 3,894.00 |
| 66 | 28-41 | 0 | 0,0339 | 1.300 | 39,283.00 |
| 67 | 28-43 | 0 | 0,0406 | 1200 | 46,701.00 |
| 68 | 31-41 | 0 | 0,0278 | 1.500 | 32,632.00 |
| 69 | 32-41 | 0 | 0,0309 | 1.400 | 35,957.00 |
| 70 | 41-43 | 0 | 0,0139 | 2.000 | 17,284.00 |
| 71 | 40-45 | 0 | 0,2205 | 180 | 13,994.00 |
| 72 | 15-16 | 0 | 0,0125 | 600 | 8,178.00 |
| 73 | 46-11 | 0 | 0,0125 | 600 | 8,178.00 |
| 74 | 24-25 | 0 | 0,0125 | 600 | 8,178.00 |
| 75 | 29-30 | 0 | 0,0125 | 600 | 8,178.00 |
| 76 | 40-41 | 0 | 0,0125 | 600 | 8,178.00 |
| 77 | 02-03 | 0 | 0,0125 | 600 | 8,178.00 |
| 78 | 05-06 | 0 | 0,0125 | 600 | 8,178.00 |
| 79 | 09-10 | 0 | 0,0125 | 600 | 8,178.00 |

## B.3 Sistema Norte-Nordeste Brasileiro (87 Barras/179 Ramos)

Tabela 43: Dados de barras - Norte-Nordeste brasileiro

| Barra | Geração em 2002 (MW) | Carga em 2002 (MW) | Geração em 2008 (MW) | Carga em 2008 (MW) |
|---|---|---|---|---|
| 1 | 0 | 1.857 | 0 | 2.747 |
| 2 | 4.048 | 0 | 4.550 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 517 | 0 | 6.422 | 0 |
| 5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 |
| 7 | 0 | 31 | 0 | 31 |
| 8 | 403 | 0 | 82 | 0 |
| 9 | 465 | 0 | 465 | 0 |
| 10 | 538 | 0 | 538 | 0 |
| 11 | 2.200 | 0 | 2.260 | 0 |
| 12 | 2.257 | 0 | 4.312 | 0 |
| 13 | 4.510 | 0 | 5.900 | 0 |
| 14 | 542 | 0 | 542 | 0 |
| 15 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 |
| 19 | 0 | 86 | 0 | 125 |
| 20 | 0 | 125 | 0 | 181 |
| 21 | 0 | 722 | 0 | 1.044 |
| 22 | 0 | 291 | 0 | 446 |
| 23 | 0 | 58 | 0 | 84 |
| 24 | 0 | 159 | 0 | 230 |
| 25 | 0 | 1.502 | 0 | 2.273 |
| 26 | 0 | 47 | 0 | 68 |
| 27 | 0 | 378 | 0 | 546 |
| 28 | 0 | 189 | 0 | 273 |

continua na próxima página

Tabela 43: Dados de barras - Norte-Nordeste brasileiro
(continuação)

| Barra | Geração em 2002 (MW) | Carga em 2002 (MW) | Geração em 2008 (MW) | Carga em 2008 (MW) |
|---|---|---|---|---|
| 29 | 0 | 47 | 0 | 68 |
| 30 | 0 | 189 | 0 | 273 |
| 31 | 0 | 110 | 0 | 225 |
| 32 | 0 | 0 | 0 | 0 |
| 33 | 0 | 0 | 0 | 0 |
| 34 | 0 | 28 | 0 | 107 |
| 35 | 1.635 | 0 | 1.531 | 0 |
| 36 | 0 | 225 | 0 | 325 |
| 37 | 169 | 0 | 114 | 0 |
| 38 | 0 | 0 | 0 | 0 |
| 39 | 0 | 186 | 0 | 269 |
| 40 | 0 | 1.201 | 0 | 1.738 |
| 41 | 0 | 520 | 0 | 752 |
| 42 | 0 | 341 | 0 | 494 |
| 43 | 0 | 0 | 0 | 0 |
| 44 | 0 | 4.022 | 0 | 5.819 |
| 45 | 0 | 0 | 0 | 0 |
| 46 | 0 | 205 | 0 | 297 |
| 47 | 0 | 0 | 0 | 0 |
| 48 | 0 | 347 | 0 | 432 |
| 49 | 0 | 777 | 0 | 1.124 |
| 50 | 0 | 5.189 | 0 | 7.628 |
| 51 | 0 | 290 | 0 | 420 |
| 52 | 0 | 707 | 0 | 1.024 |
| 53 | 0 | 0 | 0 | 0 |
| 54 | 0 | 0 | 0 | 0 |
| 55 | 0 | 0 | 0 | 0 |
| 56 | 0 | 0 | 0 | 0 |
| 57 | 0 | 0 | 0 | 0 |
| 58 | 0 | 0 | 0 | 0 |

continua na próxima página

Tabela 43: Dados de barras - Norte-Nordeste brasileiro
(continuação)

| Barra | Geração em 2002 (MW) | Carga em 2002 (MW) | Geração em 2008 (MW) | Carga em 2008 (MW) |
|---|---|---|---|---|
| 59 | 0 | 0 | 0 | 0 |
| 60 | 0 | 0 | 0 | 0 |
| 61 | 0 | 0 | 0 | 0 |
| 62 | 0 | 0 | 0 | 0 |
| 63 | 0 | 0 | 0 | 0 |
| 64 | 0 | 0 | 0 | 0 |
| 65 | 0 | 0 | 0 | 0 |
| 66 | 0 | 0 | 0 | 0 |
| 67 | 1.242 | 0 | 1.242 | 0 |
| 68 | 888 | 0 | 888 | 0 |
| 69 | 902 | 0 | 902 | 0 |
| 70 | 0 | 0 | 0 | 0 |
| 71 | 0 | 0 | 0 | 0 |
| 72 | 0 | 0 | 0 | 0 |
| 73 | 0 | 0 | 0 | 0 |
| 74 | 0 | 0 | 0 | 0 |
| 75 | 0 | 0 | 0 | 0 |
| 76 | 0 | 0 | 0 | 0 |
| 77 | 0 | 0 | 0 | 0 |
| 78 | 0 | 0 | 0 | 0 |
| 79 | 0 | 0 | 0 | 0 |
| 80 | 0 | 0 | 0 | 0 |
| 81 | 0 | 0 | 0 | 0 |
| 82 | 0 | 0 | 0 | 0 |
| 83 | 0 | 0 | 0 | 0 |
| 84 | 0 | 0 | 0 | 0 |
| 85 | 0 | 487 | 0 | 705 |
| 86 | 0 | 0 | 0 | 0 |
| 87 | 0 | 0 | 0 | 0 |
| Total | 20.316 | 20.316 | 29.748 | 29.748 |

Tabela 44: Dados de linhas - Norte-Nordeste brasileiro

| Nº | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 1 | 01-02 | 2 | 0,0374 | 1.000 | 44,056.00 |
| 2 | 02-04 | 0 | 0,0406 | 1.000 | 48,880.00 |
| 3 | 02-60 | 0 | 0,0435 | 1.000 | 52,230.00 |
| 4 | 02-87 | 1 | 0,0259 | 1.000 | 31,192.00 |
| 5 | 03-71 | 0 | 0,0078 | 3.200 | 92,253.00 |
| 6 | 03-81 | 0 | 0,0049 | 3.200 | 60,153.00 |
| 7 | 03-83 | 0 | 0,0043 | 3.200 | 53,253.00 |
| 8 | 03-87 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 9 | 04-05 | 1 | 0,0435 | 1.000 | 52,230.00 |
| 10 | 04-06 | 0 | 0,0487 | 1.000 | 58,260.00 |
| 11 | 04-32 | 0 | 0,0233 | 300 | 7,510.00 |
| 12 | 04-60 | 0 | 0,0215 | 1.000 | 26,770.00 |
| 13 | 04-68 | 0 | 0,0070 | 1.000 | 10,020.00 |
| 14 | 04-69 | 0 | 0,0162 | 1.000 | 20,740.00 |
| 15 | 04-81 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 16 | 04-87 | 1 | 0,0218 | 1.000 | 26,502.00 |
| 17 | 05-06 | 1 | 0,0241 | 1.000 | 29,852.00 |
| 18 | 05-38 | 2 | 0,0117 | 600 | 8,926.00 |
| 19 | 05-56 | 0 | 0,0235 | 1.000 | 29,182.00 |
| 20 | 05-58 | 0 | 0,0220 | 1.000 | 27,440.00 |
| 21 | 05-60 | 0 | 0,0261 | 1.000 | 32,130.00 |
| 22 | 05-68 | 0 | 0,0406 | 1.000 | 48,880.00 |
| 23 | 05-70 | 0 | 0,0464 | 1.000 | 55,580.00 |
| 24 | 05-80 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 25 | 06-07 | 1 | 0,0288 | 1.000 | 35,212.00 |
| 26 | 06-37 | 1 | 0,0233 | 300 | 7,510.00 |
| 27 | 06-67 | 0 | 0,0464 | 1.000 | 55,580.00 |
| 28 | 06-68 | 0 | 0,0476 | 1.000 | 5,6920.00 |
| 29 | 06-70 | 0 | 0,0371 | 1.000 | 44,860.00 |
| 30 | 06-75 | 0 | 0,0058 | 1.200 | 21232.00 |
| 31 | 07-08 | 1 | 0,0234 | 1.000 | 29,048.00 |

continua na próxima página

Tabela 44: Dados de linhas - Norte-Nordeste brasileiro
(continuação)

| Nº | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 32 | 07-53 | 0 | 0,0452 | 1.000 | 54,240.00 |
| 33 | 07-62 | 0 | 0,0255 | 1.000 | 31,460.00 |
| 34 | 08-09 | 1 | 0,0186 | 1.000 | 23,420.00 |
| 35 | 08-12 | 0 | 0,0394 | 1.000 | 47,540.00 |
| 36 | 08-17 | 0 | 0,0447 | 1.000 | 53,570.00 |
| 37 | 08-53 | 1 | 0,0365 | 1.200 | 44,190.00 |
| 38 | 08-62 | 0 | 0,0429 | 1.000 | 51,560.00 |
| 39 | 08-73 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 40 | 09-10 | 1 | 0,0046 | 1.000 | 7,340.00 |
| 41 | 10-11 | 1 | 0,0133 | 1.000 | 17,390.00 |
| 42 | 11-12 | 1 | 0,0041 | 1.200 | 6,670.00 |
| 43 | 11-15 | 1 | 0,0297 | 1.200 | 36,284.00 |
| 44 | 11-17 | 1 | 0,0286 | 1.200 | 35,078.00 |
| 45 | 11-53 | 1 | 0,0254 | 1.000 | 31,326.00 |
| 46 | 12-13 | 1 | 0,0046 | 1.200 | 7,340.00 |
| 47 | 12-15 | 1 | 0,0256 | 1.200 | 31,594.00 |
| 48 | 12-17 | 1 | 0,0246 | 1.200 | 30,388.00 |
| 49 | 12-35 | 2 | 0,0117 | 600 | 8,926.00 |
| 50 | 12-84 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 51 | 13-14 | 0 | 0,0075 | 1.200 | 10,690.00 |
| 52 | 13-15 | 0 | 0,0215 | 1.200 | 26,770.00 |
| 53 | 13-17 | 0 | 0,0232 | 1.200 | 28,780.00 |
| 54 | 13-45 | 1 | 0,0290 | 1.200 | 35,480.00 |
| 55 | 13-59 | 1 | 0,0232 | 1.200 | 28,780.00 |
| 56 | 14-17 | 0 | 0,0232 | 1.200 | 28,780.00 |
| 57 | 14-45 | 0 | 0,0232 | 1.200 | 28,780.00 |
| 58 | 14-59 | 0 | 0,0157 | 1.200 | 20,070.00 |
| 59 | 15-16 | 2 | 0,0197 | 1.200 | 24,760.00 |
| 60 | 15-45 | 0 | 0,0103 | 1.200 | 13,906.00 |
| 61 | 15-46 | 1 | 0,0117 | 600 | 8,926.00 |

continua na próxima página

Tabela 44: Dados de linhas - Norte-Nordeste brasileiro
(continuação)

| Nº | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 62 | 15-53 | 0 | 0,0423 | 1.000 | 50,890.00 |
| 63 | 16-44 | 4 | 0,0117 | 600 | 8,926.00 |
| 64 | 16-45 | 0 | 0,0220 | 1.200 | 27,440.00 |
| 65 | 16-61 | 0 | 0,0128 | 1.000 | 16,720.00 |
| 66 | 16-77 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 67 | 17-18 | 2 | 0,0170 | 1.200 | 21,678.00 |
| 68 | 17-59 | 0 | 0,0170 | 1200 | 21,678.00 |
| 69 | 18-50 | 4 | 0,0117 | 600 | 8,926.00 |
| 70 | 18-59 | 1 | 0,0331 | 1.200 | 40,170.00 |
| 71 | 18-74 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 72 | 19-20 | 1 | 0,0934 | 170 | 5,885.00 |
| 73 | 19-22 | 1 | 0,1877 | 170 | 11,165.00 |
| 74 | 20-21 | 1 | 0,0715 | 300 | 6,960.00 |
| 75 | 20-38 | 2 | 0,1382 | 300 | 12,840.00 |
| 76 | 20-56 | 0 | 0,0117 | 600 | 8,926.00 |
| 77 | 20-66 | 0 | 0,2064 | 170 | 12,210.00 |
| 78 | 21-57 | 0 | 0,0117 | 600 | 8,926.00 |
| 79 | 22-23 | 1 | 0,1514 | 170 | 9,130.00 |
| 80 | 22-37 | 2 | 0,2015 | 170 | 11,935.00 |
| 81 | 22-58 | 0 | 0,0233 | 300 | 7,510.00 |
| 82 | 23-24 | 1 | 0,1651 | 170 | 9,900.00 |
| 83 | 24-25 | 1 | 0,2153 | 170 | 12,705.00 |
| 84 | 24-43 | 0 | 0,0233 | 300 | 7,510.00 |
| 85 | 25-26 | 2 | 0,1073 | 300 | 29,636.00 |
| 86 | 25-55 | 0 | 0,0117 | 600 | 8,926.00 |
| 87 | 26-27 | 2 | 0,1404 | 300 | 25,500.00 |
| 88 | 26-29 | 1 | 0,1081 | 170 | 6,710.00 |
| 89 | 26-54 | 0 | 0,0117 | 600 | 8,926.00 |
| 90 | 27-28 | 3 | 0,0826 | 170 | 5,335.00 |
| 91 | 27-35 | 2 | 0,1367 | 300 | 25,000.00 |

continua na próxima página

Tabela 44: Dados de linhas - Norte-Nordeste brasileiro
(continuação)

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 92 | 27-53 | 1 | 0,0117 | 600 | 8,926.00 |
| 93 | 28-35 | 3 | 0,1671 | 170 | 9,900.00 |
| 94 | 29-30 | 1 | 0,0688 | 170 | 4,510.00 |
| 95 | 30-31 | 1 | 0,0639 | 170 | 4,235.00 |
| 96 | 30-63 | 0 | 0,0233 | 300 | 7510.00 |
| 97 | 31-34 | 1 | 0,1406 | 170 | 8,525.00 |
| 98 | 32-33 | 0 | 0,1966 | 170 | 11,660.00 |
| 99 | 33-67 | 0 | 0,0233 | 300 | 7,510.00 |
| 100 | 34-39 | 2 | 0,1160 | 170 | 7,150.00 |
| 101 | 34-41 | 2 | 0,0993 | 170 | 6,215.00 |
| 102 | 35-46 | 4 | 0,2172 | 170 | 12,705.00 |
| 103 | 35-47 | 2 | 0,1327 | 170 | 8,085.00 |
| 104 | 35-51 | 3 | 0,1602 | 170 | 9,625.00 |
| 105 | 36-39 | 2 | 0,1189 | 170 | 7,315.00 |
| 106 | 36-46 | 2 | 0,0639 | 170 | 4,235.00 |
| 107 | 39-42 | 1 | 0,0973 | 170 | 6,105.00 |
| 108 | 39-86 | 0 | 0,0233 | 300 | 7,510.00 |
| 109 | 40-45 | 1 | 0,0117 | 600 | 8,926.00 |
| 110 | 40-46 | 3 | 0,0875 | 170 | 5,500.00 |
| 111 | 41-64 | 0 | 0,0233 | 300 | 7,510.00 |
| 112 | 42-44 | 2 | 0,0698 | 170 | 4,565.00 |
| 113 | 42-85 | 2 | 0,0501 | 170 | 3,465.00 |
| 114 | 43-55 | 0 | 0,0254 | 1.000 | 31,326.00 |
| 115 | 43-58 | 0 | 0,0313 | 1.000 | 38,160.00 |
| 116 | 44-46 | 3 | 0,1671 | 170 | 10,010.00 |
| 117 | 47-48 | 2 | 0,1966 | 170 | 11,660.00 |
| 118 | 48-49 | 1 | 0,0757 | 170 | 4,895.00 |
| 119 | 48-50 | 2 | 0,0256 | 170 | 2,090.00 |
| 120 | 48-51 | 2 | 0,2163 | 170 | 12,760.00 |
| 121 | 49-50 | 1 | 0,0835 | 170 | 5,335.00 |

continua na próxima página

Tabela 44: Dados de linhas - Norte-Nordeste brasileiro
(continuação)

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 122 | 51-52 | 2 | 0,0560 | 170 | 3,795.00 |
| 123 | 52-59 | 1 | 0,0117 | 600 | 8,926.00 |
| 124 | 53-54 | 0 | 0,0270 | 1.000 | 32,120.00 |
| 125 | 53-70 | 0 | 0,0371 | 1.000 | 44,860.00 |
| 126 | 53-76 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 127 | 53-86 | 0 | 0,0389 | 1000 | 46,870.00 |
| 128 | 54-55 | 0 | 0,0206 | 1.000 | 25,028.00 |
| 129 | 54-58 | 0 | 0,0510 | 1000 | 60,940.00 |
| 130 | 54-63 | 0 | 0,0203 | 1.000 | 25,430.00 |
| 131 | 54-70 | 0 | 0,0360 | 1000 | 43,520.00 |
| 132 | 54-79 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 133 | 56-57 | 0 | 0,0122 | 1000 | 16,050.00 |
| 134 | 58-78 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 135 | 60-66 | 0 | 0,0233 | 300 | 7,510.00 |
| 136 | 60-87 | 0 | 0,0377 | 1.000 | 45,530.00 |
| 137 | 61-64 | 0 | 0,0186 | 1.000 | 23,420.00 |
| 138 | 61-85 | 0 | 0,0233 | 300 | 7,510.00 |
| 139 | 61-86 | 0 | 0,0139 | 1.000 | 18,060.00 |
| 140 | 62-67 | 0 | 0,0464 | 1.000 | 55,580.00 |
| 141 | 62-68 | 0 | 0,0557 | 1.000 | 66,300.00 |
| 142 | 62-72 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 143 | 63-64 | 0 | 0,0290 | 1.000 | 35,480.00 |
| 144 | 65-66 | 0 | 0,3146 | 170 | 18,260.00 |
| 145 | 65-87 | 0 | 0,0233 | 300 | 7,510.00 |
| 146 | 67-68 | 0 | 0,0290 | 1.000 | 35,480.00 |
| 147 | 67-69 | 0 | 0,0209 | 1.000 | 26,100.00 |
| 148 | 67-71 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 149 | 68-69 | 0 | 0,0139 | 1.000 | 18,060.00 |
| 150 | 68-83 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 151 | 68-87 | 0 | 0,0186 | 1.000 | 23,240.00 |

continua na próxima página

Tabela 44: Dados de linhas - Norte-Nordeste brasileiro
(continuação)

| $N^o$ | Ramo | Linhas existentes | Reatância (pu) | Capacidade (MW) | Custo $10^3$ US$ |
|---|---|---|---|---|---|
| 152 | 69-87 | 0 | 0,0139 | 1.000 | 18,060.00 |
| 153 | 70-82 | 0 | 0,0058 | 1.200 | 21,232.00 |
| 154 | 71-72 | 0 | 0,0108 | 3.200 | 125,253.00 |
| 155 | 71-75 | 0 | 0,0108 | 3.200 | 125,253.00 |
| 156 | 71-83 | 0 | 0,0067 | 3.200 | 80,253.00 |
| 157 | 72-73 | 0 | 0,0100 | 3.200 | 116,253.00 |
| 158 | 72-83 | 0 | 0,0130 | 3.200 | 149,253.00 |
| 159 | 73-74 | 0 | 0,0130 | 3.200 | 149,253.00 |
| 160 | 73-75 | 0 | 0,0130 | 3.200 | 149,253.00 |
| 161 | 73-84 | 0 | 0,0092 | 3.200 | 107,253.00 |
| 162 | 74-84 | 0 | 0,0108 | 3.200 | 125,253.00 |
| 163 | 75-76 | 0 | 0,0162 | 3.200 | 185,253.00 |
| 164 | 75-81 | 0 | 0,0113 | 3.200 | 131,253.00 |
| 165 | 75-82 | 0 | 0,0086 | 3.200 | 101,253.00 |
| 166 | 75-83 | 0 | 0,0111 | 3.200 | 128,253.00 |
| 167 | 76-77 | 0 | 0,0130 | 3.200 | 149,253.00 |
| 168 | 76-82 | 0 | 0,0086 | 3.200 | 101,253.00 |
| 169 | 76-84 | 0 | 0,0059 | 3.200 | 70,953.00 |
| 170 | 77-79 | 0 | 0,0151 | 3.200 | 173,253.00 |
| 171 | 77-84 | 0 | 0,0115 | 3.200 | 132,753.00 |
| 172 | 78-79 | 0 | 0,0119 | 3.200 | 137,253.00 |
| 173 | 78-80 | 0 | 0,0051 | 3.200 | 62,253.00 |
| 174 | 79-82 | 0 | 0,0084 | 3.200 | 98,253.00 |
| 175 | 80-81 | 0 | 0,0101 | 3.200 | 117,753.00 |
| 176 | 80-82 | 0 | 0,0108 | 3.200 | 125,253.00 |
| 177 | 80-83 | 0 | 0,0094 | 3.200 | 110,253.00 |
| 178 | 81-83 | 0 | 0,0016 | 3.200 | 23,253.00 |
| 179 | 82-84 | 0 | 0,0135 | 3.200 | 155,253.00 |

<table>
<tr><td colspan="2">Classificação de Segurança<br>Livre</td><td colspan="2">Documento no.</td></tr>
<tr><td colspan="2">Data Fev/2008</td><td colspan="2">Projeto no.</td></tr>
<tr><td colspan="3">Título e subtítulo<br>ALGORITMOS BUSCA TABU PARALELOS APLICADOS AO PLANEJAMENTO DA EXPANSÃO DA TRANSMISSÃO DE ENERGIA ELÉTRICA</td><td>No. do volume<br>No. da parte</td></tr>
<tr><td colspan="4">Título do projeto</td></tr>
<tr><td colspan="2">Entidade Executora (autor coletivo)</td><td colspan="2">Autor(es)<br>Elisângela Menegasso Mansano</td></tr>
<tr><td colspan="4">Entidade patrocinada (cliente ou destinatário principal)<br>PPGEE - UNESP Campus de Ilha Solteira</td></tr>
<tr><td colspan="4">Resumo<br>Neste trabalho, aborda-se o uso da metaheurística Busca Tabu (BT) na resolução do Problema de Planejamento da Expansão da Transmissão (PPET), analisado sob o ponto de vista estático, com o desenvolvimento de algoritmos paralelos em ambiente MPI (“Message Passing Interface”). A metaheurística Busca Tabu, é uma técnica baseada em parâmetros de controle, a estrutura de vizinhança e seu próprio algoritmo com poderosas estratégias de busca. Nesta técnica, dada uma configuração, deseja-se passar ao melhor vizinho através da entrada e saída de ramos, obtendo assim a configuração incumbente. Com essa configuração que é considerada como a melhor configuração encontrada até o momento, e mesmo sendo um bom valor o sistema continua a busca procurando mais configurações até encontrar uma que seja melhor que as já encontradas até o momento. As versões paralelas dos algoritmos foram desenvolvidas a partir de um algoritmo BT serial avançado, sob o paradigma de programação SPMD (“Single Program, Multiple Data”), e as mesmas foram testadas para sistemas testes de pequeno porte (Garver - 6 barras/15 ramos), médio porte (Sul brasileiro - 46 barras/79 ramos) e grande porte (Norte-Nordeste brasileiro - 87 barras/179 ramos) e seus resultados comparados com o resultado do algoritmo BT serial. Esta comparação mostrou que os algoritmos propostos obtiveram um melhor desempenho, com alta eficiência.</td></tr>
<tr><td colspan="4">Palavras-chave<br>Metaheurística, Planejamento da Expansão de Redes, Busca Tabu, Programação Paralela.</td></tr>
<tr><td>No. da edição</td><td>No. de páginas<br>144</td><td>ISSN (para relatórios publicados)</td><td>Classificação (CDC ou CDD)</td></tr>
<tr><td colspan="2">Distribuidor</td><td>Número de exemplares<br>2 (dois)</td><td>Preço<br>-</td></tr>
<tr><td colspan="4">Observações<br>Trabalho apresentado ao Programa de Pós-Graduação em Engenharia Elétrica, UNESP - Campus de Ilha Solteira, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.</td></tr>
</table>