

Frederico Rodrigues Borges da Cruz

Algoritmos para Problemas de Planejamento de Redes

Tese apresentada ao Departamento de
Ciência da Computação do Instituto de
Ciências Exatas da Universidade Federal
de Minas Gerais, como requisito parcial à
obtenção do título de Doutor em Ciência
da Computação.

Universidade Federal de Minas Gerais
Belo Horizonte, fevereiro de 1997

© 1997, Frederico Rodrigues Borges da Cruz.
Todos os direitos reservados.

Cruz, Frederico Rodrigues Borges da.	
C957a	Algoritmos para problemas de planejamento de redes [manuscrito] / Frederico Rodrigues Borges da Cruz. – 1997. x, 80 f. il.
	Orientador: Geraldo Robson Mateus. Tese (Doutorado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. Referências: f.75-80.
	1. Computação – Teses. 2. Algoritmos de computador – Teses. 3. Redes de computadores – Planejamento -Teses. 4. Otimização combinatoria – Teses. I. Mateus, Geraldo Robson. II. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.
	CDU 519.6*62(043)

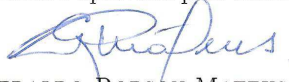
Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6ª Região nº 1510

FOLHA DE APROVAÇÃO

Algoritmos para Problemas de Planejamento de Redes

FREDERICO RODRIGUES BORGES DA CRUZ

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:



PROF. GERALDO ROBSON MATEUS - Orientador
DCC - ICEX - UFMG



PROF. HENRIQUE PACCA LOUREIRO LUNA
DCC - ICEX - UFMG



PROF. NÍVIO ZIVIANI
DCC - ICEX - UFMG



PROF. NELSON MACULAN FILHO
COPPE - UFRJ



PROF. JAMES MACGREGOR SMITH
Univ. of Massachusetts at Amherst - EUA

Belo Horizonte, 04 de fevereiro de 1997.

*Dedico este trabalho à Carolina
e ao saudoso amigo Alpheu.*

Agradecimentos

Gostaria de deixar expressos aqui os meus mais sinceros agradecimentos a todos aqueles que de alguma forma contribuíram para que este trabalho chegasse a um bom termo.

Em especial, agradeço à Professora Ana e ao Professor Borges, pelas primeiras lições.

Agradeço também ao Professor Robson, pela pronta disposição em orientar-me neste trabalho e também pela imensa paciência e atenção dispensadas.

Ao Professor Smith, da *University of Massachusetts at Amherst*, EUA, que tão bem me acolheu durante parte do desenvolvimento deste trabalho, fica minha gratidão.

Ao Professor Pacca e ao Professor Nelson, vão os meus agradecimentos, pelas palavras de incentivo e proveitosas discussões.

Ao Professor Nívio, agradeço pelas contribuições e pela prontidão em aceitar participar da banca examinadora deste trabalho.

Ao Departamento de Ciência da Computação, o meu muito obrigado, pela confiança depositada.

Agradecimentos também vão à CAPES e ao CNPq, pelo suporte financeiro.

E um agradecimento bem especial vai à Professora Nely, que vem gentilmente revendo este trabalho desde os primeiros manuscritos ...

Sumário

Lista de Figuras	xi
Lista de Tabelas	xiii
Resumo	xv
Summary	xvii
1 Introdução	1
1.1 Escopo da Tese	3
1.2 Contribuições da Tese	3
1.3 Organização da Tese	4
2 O Problema Não-Capacitado de Fluxos com Custos Fixos nos Arcos	7
2.1 Apresentação do Problema	7
2.2 Formulação Matemática	9
2.3 Método de Solução	11
2.3.1 Cálculo dos Limites L e U	12
2.3.2 Estratégias de <i>Branching</i>	15
2.3.3 Redução do Problema	17
2.3.4 Projeto das Estruturas de Dados	18
2.4 Resultados Experimentais	20
2.5 Conclusões	23
3 O Problema de Planejamento de Redes em Multi-Níveis	25
3.1 Introdução	25
3.2 Formulações	27
3.3 Algoritmo <i>Branch-and-Bound</i>	30
3.3.1 Cálculo dos Limites L e U	32
3.3.2 Escolha das Variáveis para <i>Branching</i>	36
3.3.3 Algoritmo de Redução	39
3.3.4 Definição das Estruturas de Dados	39
3.4 Experiência Computacional	44
3.5 Conclusões	46

4	Experiência Computacional com Implementações Paralelas do Algoritmo	49
	<i>Branch-and-Bound</i>	
4.1	Introdução	49
4.2	Algoritmos Implementados	50
	4.2.1 Versão Centralizada	51
	4.2.2 Versão Distribuída	52
4.3	Apresentação e Discussão dos Resultados	53
4.4	Observações Finais	57
5	Propostas de Continuidade	61
5.1	Heurísticas	61
5.2	Limites Inferiores	61
	5.2.1 Métodos Duais	62
	5.2.2 Formulações Alternativas	62
	5.2.3 Restrições Adicionais	64
	5.2.4 Teoria das Inequações Válidas	68
5.3	Testes de Redução	68
5.4	Conclusões	68
6	Conclusões	69
A	Fragmentos de Código em C	71
	Bibliografia	75

Lista de Figuras

1.1	Uma Rede em Multi-Níveis	2
2.1	Uma Rede com Fonte Única	8
2.2	Algoritmo <i>Branch-and-Bound</i> para o Problema NCFCF	11
2.3	Algoritmo de Otimização por Subgradientes	14
2.4	Estratégia Ingênua de <i>Branching</i> para o Problema NCFCF	15
2.5	Estratégia Melhorada de <i>Branching</i> para o Problema NCFCF	16
2.6	Teste de Redução para o Problema NCFCF	17
2.7	Procedimento de Restabelecimento para o Problema NCFCF	18
2.8	Representação do Dígrafo $\mathcal{D} = (N, A)$ por Listas de Adjacências	19
2.9	Representação do Conjunto S por Listas Encadeadas	19
3.1	Algoritmo <i>Branch-and-Bound</i> para o Problema PRMN	31
3.2	Algoritmo para Resolução do Problema (L_1)	35
3.3	Estratégia Simplificada de <i>Branching</i> para o Problema PRMN	37
3.4	Estratégia Melhorada de <i>Branching</i> para o Problema PRMN	38
3.5	Algoritmo de Redução para o Problema PRMN	40
3.6	Procedimento de Restabelecimento para o Problema PRMN	41
3.7	Representação do Dígrafo $\mathcal{D} = (N, A)$ por Listas de Adjacências Multi-Ponderadas	42
3.8	Representação dos Conjuntos R^l e D^l , $l = 1, 2, \dots, m$, por Listas Encadeadas Simples	43
4.1	Algoritmo <i>Branch-and-Bound</i> Não-Recursivo	50
4.2	Implementação Centralizada do Algoritmo <i>Branch-and-Bound</i>	51
4.3	Implementação Distribuída do Algoritmo <i>Branch-and-Bound</i>	52
4.4	<i>Speedup</i> Médio para as Implementações Paralelas	55
4.5	Diagramas de Kiviat para a Versão Centralizada	56
4.6	Diagramas de Kiviat para a Versão Distribuída Estática	57
4.7	Árvores de Pesquisa para a Versão Distribuída Estática	58
4.8	Diagramas de Kiviat para a Versão Distribuída de Karp-Zhang	59
4.9	Diagramas de Kiviat para a Versão Distribuída de Karp-Zhang Modificada	59
5.1	Rede Exemplo com $ N = 18$ e $ A = 54$	66

A.1	Fragmento de Código em C para Construção de Listas de Adjacências . .	72
A.2	Fragmento de Código em C para Construção de Listas Encadeadas Simples	73

Lista de Tabelas

2.1	Resultados Computacionais para Redes Aleatórias, com $ N = 16$ e $ N = 32$	21
2.2	Resultados Computacionais para as Redes de Beasley, (Beasley, 1990), com $f_{ij}/\Omega_{ij} = 1$ e $c_{ij}/\Omega_{ij} = 10$	22
3.1	Efeito da Estratégia de <i>Branching</i> no Problema PRMN	45
3.2	Ganho do <i>Branch-and-Bound</i> sobre a Enumeração Explícita no Problema PRMN	46
3.3	Efeito do Algoritmo de Redução no Problema PRMN	47
3.4	Soluções no Primeiro Nó para o Problema PRMN	48
4.1	Detalhes de <i>Hardware</i> das Máquinas Utilizadas	54
4.2	Tempos Sequenciais Básicos	55
5.1	Qualidade da Relaxação Linear <i>versus</i> Relaxação Lagrangeana	62
5.2	Qualidade das Relaxações Lineares para a Razão $f_{ij}/c_{ij} = 0.1$	64
5.3	Qualidade das Relaxações Lineares para a Razão $f_{ij}c_{ij} = 10$	65
5.4	Resultados Computacionais para a Rede Exemplo	67

Resumo

Nessa tese, estudamos alguns problemas de planejamento de redes (*network design problems*), uma denominação genérica que agrupa muitos problemas importantes. São problemas definidos em grafos, onde é procurado um subconjunto de arcos e de nós, cumprindo certos requisitos. Apresentamos o problema não-capacitado de fluxos com custos fixos (NCFCF), com uma revisão bibliográfica atualizada sobre os avanços no estudo do problema, e desenvolvemos um método de solução original. Mostramos resultados computacionais, onde são resolvidos problemas conhecidos na literatura. Introduzimos um novo modelo, o problema de planejamento de redes em multi-níveis (PRMN), apresentando uma revisão bibliográfica sobre problemas correlatos e mostrando o crescente interesse nesse tipo de modelo. Os métodos desenvolvidos para o problema NCFCF são então estendidos a esse novo modelo, o problema PRMN, com resultados promissores, conforme atestam os resultados computacionais que apresentamos. Elaboramos um estudo comparativo entre diversas implementações paralelas para os algoritmos desenvolvidos, com resultados práticos encorajadores, em termos de *speedup*. Fazemos uma revisão bibliográfica e discutimos possíveis extensões para o nosso trabalho. Finalmente, é importante reforçar que os principais resultados aqui apresentados podem ser estendidos a muitos outros problemas de planejamento de redes.

Summary

In this thesis, a special class of network design problems representing an important collection of mixed-integer programming problems is studied. The problems are defined on a digraph, where an optimal subset of arcs and nodes fulfilling a special set of constraints is sought. The uncapacitated fixed-charge network flow (UFNF) problem is presented along with a comprehensive review of the research advances in the field. An original solution algorithm is developed and its practical efficiency is demonstrated through a comprehensive set of computational experiments. The multi-level network design (MLND) problem is next developed and a review of the research efforts on similar problems is presented showing the increasing attention such models have recently received. The methods developed for the UFNF problem are extended to the MLND problem, achieving encouraging computational results. Also, parallel implementations for the algorithms are developed, demonstrating that this is a promising area for future investigations. Some possible research directions for further study on the MLND problem are proposed and briefly outlined. Finally, it is noteworthy to point out that the main results in this thesis may be extended to many other network design problems.

Capítulo 1

Introdução

Sob a denominação genérica de “problemas de planejamento de redes” (*network design problems*) são agrupados muitos problemas práticos importantes. Tais problemas são definidos em grafos onde é procurado um subconjunto de arcos e de nós de tal forma a cumprir certos requisitos. Mudando tais requisitos, diferentes problemas emergem. O objeto de nossa pesquisa têm sido os problemas de planejamento de redes em multi-níveis (PRMN), bem como um dos seus casos particulares, *i.e.* o problema não-capacitado de fluxos com custos fixos nos arcos (NCFCF). Uma rede em multi-níveis está sendo representada na Figura 1.1. A formidável riqueza de detalhes presentes nas redes em multi-níveis possibilita muitos tratamentos diferentes. Desde o início, optamos por aqueles problemas que integram no mesmo modelo aspectos de localização discreta, planejamento topológico de redes e também dimensionamento.

O estudo de redes em multi-níveis é importante em termos teóricos porque elas podem ser vistas com uma generalização de importantes problemas de otimização em redes, tais como problemas de planejamento de topologia de redes, problemas de fluxos com custos fixos ou problemas de localização não-capacitada. O estudo de algoritmos e da complexidade de tais problemas tem relevância comprovada, (Papadimitriou e Stieglitz, 1982, Tarjan, 1983, Cruz, 1991).

Outra boa razão está no forte apelo prático que tais problemas têm. Modernos sistemas de telecomunicações, sistemas de transporte e sistemas de transmissão e distribuição de energia elétrica, para citar poucos exemplos, são hierarquicamente organizados, *i.e.* possuem multi-níveis. Os fluxos de primeira, segunda e m -ésima hierarquia, Figura 1.1, poderiam representar: (i) sinais de voz, vídeo e dados, em redes de telecomunicações, (ii) carretas, caminhões e carros particulares, em redes de transporte, ou ainda (iii) vários níveis de tensão em redes de transmissão e distribuição de energia elétrica.

Essencialmente, os problemas PRMN e NCFCF são problemas de otimização em redes pertencentes à classe \mathcal{NP} -árdua, portanto, intratáveis. O propósito dessa tese é descrever um estudo extensivo do problema PRMN sob vários aspectos. O problema NCFCF é usado algumas vezes como ambiente de “laboratório” para o desenvolvimento dos algoritmos para o problema PRMN. Sucessivamente, descrevemos nesse Capítulo o escopo, as contribuições e a organização da tese.

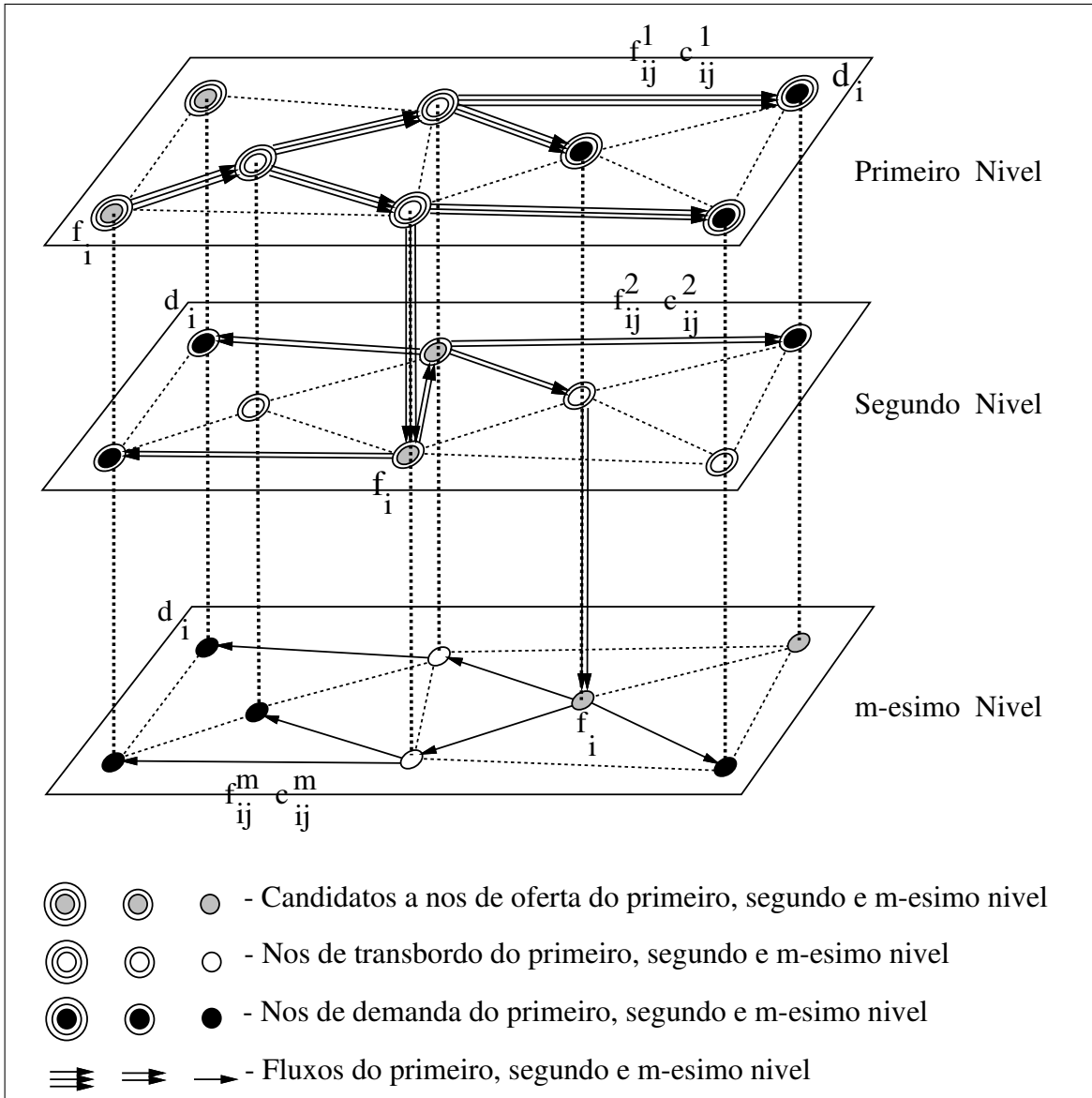


Figura 1.1: Uma Rede em Multi-Níveis

1.1 Escopo da Tese

A tese cobre alguns dos principais aspectos do estudo do problema PRMN, indo de uma análise teórica e desenvolvimento de métodos de solução, até a implementação desses métodos para uma avaliação experimental e a resolução de problemas conhecidos na literatura da área. A análise teórica inclui alguns pontos fundamentais, tais como o estudo da complexidade computacional do problema e a prova da característica em forma de árvore das soluções ótimas. O desenvolvimento dos métodos de solução concentra-se no critério de escolha das variáveis para *branching*, no contexto dos algoritmos *branch-and-bound*, e na criação de algoritmos de redução. A implementação dos métodos é na linguagem C++, e também usando um pacote para processamento paralelo e outro para solução de problemas lineares.

1.2 Contribuições da Tese

As contribuições dessa tese consistem nos seguintes aspectos:

1. Uma extensiva revisão bibliográfica sobre os avanços recentes no estudo do problema NCFCF é apresentada.
2. Uma formulação alternativa para o problema NCFCF é proposta e estudada. Essa formulação generaliza algumas formulações existentes e facilita muito a descrição dos algoritmos propostos.
3. Descrevemos um algoritmo exato (*branch-and-bound*) para resolução do problema NCFCF.
4. Uma nova estratégia de escolha da variável de *branching* é proposta para resolução do problema NCFCF sob um algoritmo *branch-and-bound*.
5. Uma nova técnica de redução baseada na relaxação lagrangeana é apresentada para o problema NCFCF.
6. Desenvolvemos um bom limite inferior para o problema NCFCF, baseado na relaxação lagrangeana, bem como uma heurística eficaz para cálculo dos limites superiores.
7. Projetamos uma eficiente estrutura de dados para representação do problema NCFCF.
8. Fazemos um importante estudo empírico sobre o uso do algoritmo de subgradientes no contexto do algoritmo *branch-and-bound*.
9. Apresentamos uma revisão bibliográfica sobre problemas correlatos ao problema PRMN, mostrando o crescente interesse por esse tipo de modelo.

10. Formulamos o problema PRMN. Não é do nosso conhecimento, na literatura da área, uma formulação desse porte, integrando os aspectos de localização discreta, planejamento topológico e dimensionamento.
11. Propomos um algoritmo exato, também do tipo *branch-and-bound*, para o problema geral PRMN.
12. Estendemos para o problema PRMN a estratégia de *branching* desenvolvida para o problema NCFCF.
13. Estendemos para o problema PRMN o teste de redução desenvolvido para o problema NCFCF.
14. Desenvolvemos um limite inferior justo para o problema PRMN, também baseado na relaxação lagrangeana, bem como uma boa heurística lagrangeana para o cálculo dos limites superiores.
15. Estendemos para o problema PRMN as estruturas de dados projetadas para o problema NCFCF.
16. Fazemos um extenso estudo empírico usando os algoritmos estendidos.
17. Fazemos um estudo comparativo entre a implementação sequencial e paralela dos algoritmos desenvolvidos para os problemas NCFCF e PRMN.
18. Apresentamos e discutimos possíveis formas de melhoria dos resultados obtidos e apontamos possíveis direções para a continuidade da pesquisa na área.

1.3 Organização da Tese

A tese é organizada do seguinte modo. No Capítulo 2, apresentamos o problema NCFCF, com uma revisão bibliográfica atualizada sobre os recentes avanços no estudo do problema. Uma formulação matemática tradicional é apresentada e uma formulação alternativa é proposta. Descrevemos o método de solução empregado, mostrando os diversos procedimentos auxiliares criados. Fazemos um estudo minucioso das suas complexidades computacionais e entramos em detalhes nas estruturas de dados empregadas e em relevantes questões que garantem a qualidade da versão implementada. Mostramos resultados computacionais, onde são resolvidos problemas conhecidos na literatura.

No Capítulo 3, tratamos do problema PRMN. Apresentamos uma revisão bibliográfica sobre alguns problemas correlatos, uma vez que o problema PRMN foi originariamente proposto por nós. Mostramos o crescente interesse nesse tipo de modelo. Formulamos o problema em duas formas distintas: uma simplificada, outra, estendida. Essa última mostrou-se mais adequada à descrição dos algoritmos de resolução, feita logo em seguida.

Fomos capazes de estender os algoritmos apresentados no Capítulo 2 e adaptá-los ao problema PRMN. Detalhes nas estruturas de dados empregadas e um estudo da complexidade computacional dos procedimentos estendidos são também apresentados. Finalmente, mostramos resultados computacionais preliminares promissores. Alcançamos um ganho excepcional sobre a enumeração explícita.

No Capítulo 4, apresentamos resultados computacionais de implementações paralelas do algoritmo *branch-and-bound* aplicado ao problema PRMN. As implementações são adequadas aos modelos de computação paralela MIMD, convenientes, portanto, para uso em redes de *workstations*. Testamos versões centralizadas e distribuídas, sendo que, para essas últimas, testamos duas estratégias de balanço de carga. Mostramos que os resultados encorajam futuros trabalhos na área, uma vez que eles apontaram para um ganho sobre o processamento sequencial, em termos de tempo total de execução.

No Capítulo 5, apresentamos e discutimos possíveis extensões para a pesquisa desenvolvida até esse ponto, apontando direções para trabalhos futuros.

Finalmente, no Capítulo 6, encerramos a tese com algumas observações finais.

Capítulo 2

O Problema Não-Capacitado de Fluxos com Custos Fixos nos Arcos

Nesse Capítulo, apresentamos o problema não-capacitado de fluxos com custos fixos nos arcos (NCFCF) e duas formulações matemáticas. Usamos como estratégia de solução o conhecido algoritmo *branch-and-bound*. Derivamos os limites superior e inferior para o problema, empregando a técnica de relaxação lagrangeana, e propomos uma eficiente estratégia de escolha das variáveis de *branching*, a qual é baseada em uma importante propriedade das soluções ótimas, que apresentamos aqui. Empregamos também a relaxação lagrangeana para desenvolvermos um algoritmo de redução. Demonstramos a eficiência prática de todos os procedimentos criados, por meio de uma ampla gama de experimentos computacionais, (Cruz *et al.*, 1995b).

2.1 Apresentação do Problema

O problema NCFCF representa uma importante classe de problemas de programação inteira mista. O problema é definido em um dígrafo $\mathcal{D} = (N, A)$, onde N é um conjunto de nós e A , um conjunto de arcos. Um dos custos envolvidos é o custo fixo pelo uso do arco. O outro, um custo variável, dependente do montante do fluxo suportado pelo arco. O objetivo é determinar uma combinação de arcos, a custo mínimo, que conduza os fluxos, dos nós de oferta a todos os nós de demanda, possivelmente passando por nós intermediários de transbordo, também conhecidos como nós de *Steiner*. Um exemplo de rede com fonte única está sendo representado na Figura 2.1.

Este é claramente um problema de otimização da classe \mathcal{NP} -árdua, uma vez que generaliza, entre outros, o problema de *Steiner* em grafos, \mathcal{NP} -árduo, (Garey e Johnson, 1979). De fato, a prova da \mathcal{NP} -complexidade do problema NCFCF já foi desenvolvida anteriormente, (Cruz *et al.*, 1994).

Esse modelo genérico tem aplicações em problemas de planejamento de redes de distribuição, de transporte e de telecomunicações. Mesmo para os casos de redes existentes, tem também aplicações nos problemas de roteamento. Além de já ser um modelo impor-

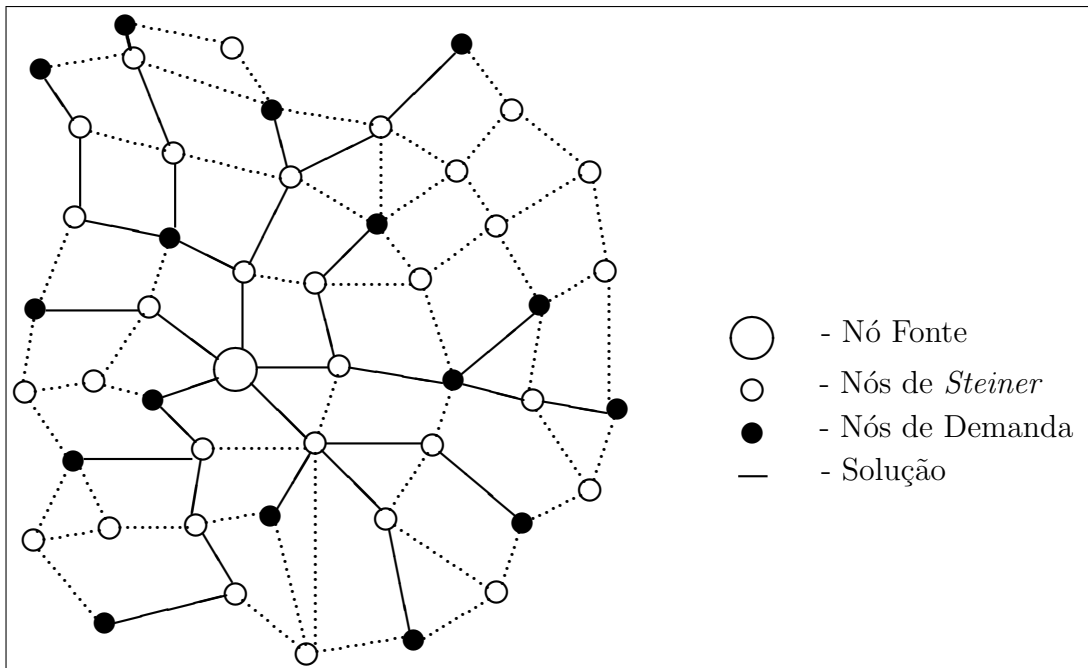


Figura 2.1: Uma Rede com Fonte Única

tante por si só, vários casos especiais do problema NCFCF são de interesse substancial. Uma maneira simples para obtenção de casos particulares é por restrição na estrutura da rede (*e.g.* o problema de transporte com custos fixos é o problema NCFCF definido sobre um grafo bipartido).

Alguns trabalhos tratando soluções exatas e aproximadas para casos particulares do problema NCFCF já foram publicados. Já foi estudado um modelo simplificado incluindo apenas os custos fixos nos arcos, aplicado à análise de um sistema coletor de gás natural, (Rothfarb *et al.*, 1970). Um modelo mais geral, aplicado a um problema de planejamento de redes locais de telecomunicações, apresentando componentes adicionais de custos fixos na função objetivo, também já foi discutido, (Luna *et al.*, 1987). Entretanto, a resolução do modelo restringiu-se a técnicas heurísticas e de otimização local. O caso particular do problema NCFCF sem nós de *Steiner* também tem sido estudado. Para esse problema, foram desenvolvidos um algoritmo exato do tipo *branch-and-bound*, combinado com decomposição de Benders, (Magnanti *et al.*, 1986), e um conjunto de heurísticas lagrangeanas, (Hochbaum e Segev, 1989). Outros casos particulares também já foram resolvidos por meio de algoritmos *branch-and-bound* associados a planos de corte fracionários (*fractional cutting-planes*), (Barr *et al.*, 1981, Cabot e Erenguc, 1984, Suhl, 1985).

Para o modelo geral do problema NCFCF, já foi introduzido um conjunto de inequações de corte (*dicut collection inequalities*), onde o problema era modelado por uma formulação multi-fluxo estendida, (Rardin e Wolsey, 1993). Também já foram desenvolvidas heurísticas do tipo *ADD* e *DROP*, (Mateus *et al.*, 1994), e um algoritmo *branch-and-bound* simplificado, (Cruz *et al.*, 1995a).

Nesse Capítulo, pretendemos resolver exatamente o problema NCFCF por meio de um algoritmo *branch-and-bound* mais elaborado. Essa estratégia enumerativa é sabida ser computacionalmente ineficiente, devido à sua complexidade de tempo, no pior caso, ser exponencial com o tamanho da entrada. Embora o algoritmo seja aceitável apenas para pequenas instâncias do problema, essa é a possibilidade mais preferida para o tratamento exato de problemas \mathcal{NP} -árduos. Entretanto, é também sabido que o uso de estratégias apropriadas de escolha das variáveis de *branching* e de técnicas de redução pode aumentar bastante o tamanho das instâncias resolvíveis em tempos razoáveis. Nesse sentido, o foco desse Capítulo é descrever um novo critério para escolha de variáveis de *branching*, um novo teste de redução e uma nova simplificação para o algoritmo *branch-and-bound*.

Na Seção 2.2, o problema NCFCF é formulado por meio de um modelo de programação matemática inteira-mista. Nessa Seção, derivamos também uma formulação alternativa. Na Seção 2.3, discutimos os métodos de solução empregados e apresentamos a estratégia para escolha da variável de *branching* e o algoritmo de redução que estamos propondo. Todos os algoritmos foram implementados e os resultados computacionais são apresentados na Seção 2.4. A Seção 2.5 encerra o Capítulo, com a apresentação de questões em aberto e algumas possíveis extensões para esse trabalho.

2.2 Formulação Matemática

Uma formulação de programação matemática inteira-mista típica, (Rardin e Wolsey, 1993), definida sobre o dígrafo $\mathcal{D} = (N, A)$, é apresentada em seguida:

(M):

$$\min \sum_{(i,j) \in A} (c_{ij}x_{ij} + f_{ij}y_{ij}), \quad (2.1)$$

s.a.:

$$\sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} = \begin{cases} - \sum_{k \in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \quad (2.2)$$

$$x_{ij} \leq \left(\sum_{k \in D} d_k \right) y_{ij}, \quad \forall (i, j) \in A, \quad (2.3)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in A, \quad (2.4)$$

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A, \quad (2.5)$$

onde N é o conjunto de nós, A é o conjunto de arcos, $\delta^+(i) = \{j \mid (i, j) \in A\}$, $\delta^-(i) = \{j \mid (j, i) \in A\}$, $s \in N$ é o nó fonte, $T \subset N$ é o conjunto de nós de transbordo, $D \subset N$ é o conjunto de nós de demanda, d_i é a demanda do nó i , f_{ij} é o custo fixo pelo uso do arco (i, j) para transmissão de fluxo e c_{ij} é o custo variável por unidade de fluxo no arco (i, j) .

A função objetivo minimiza o custo total variável, associado aos fluxos, e o custo total fixo, associado ao uso dos arcos. As restrições (2.2) garantem a conservação de fluxos nos

diversos nós. Para o nó de oferta s , a quantidade de fluxo que chega menos a que sai deve ser igual ao somatório das demandas, com o sinal trocado. Essa restrição é redundante. Nos nós de transbordo, a quantidade de fluxo que chega deve ser igual à que sai. Finalmente, nos nós de demanda, a quantidade de fluxo que chega menos a que sai deve ser igual à demanda interna do nó, d_i . As restrições (2.3) asseguram que não haja fluxo através do arco (i, j) , a menos que ele tenha sido selecionado na função objetivo.

Assumimos que o custo fixo f_{ij} seja não-negativo, pois caso contrário a variável y_{ij} poderia ser feita igual a 1 e simplesmente eliminada do problema. A variável c_{ij} é irrestrita, mas para garantirmos que a função objetivo seja limitada por baixo, assumimos que não existam circuitos com custo negativo.

Durante esse Capítulo, consideramos uma formulação estendida equivalente para o problema NCFCF. Para isso, definimos $K_0 \subseteq A$, como o conjunto de arcos rejeitados, *i.e.* $y_{ij} = 0$, $K_1 \subseteq A$, como o conjunto de arcos selecionados, *i.e.* $y_{ij} = 1$, e $K = A \setminus K_0 \setminus K_1$, como o conjunto de arcos em estado ainda indefinido. Assim, o problema NCFCF pode ser representado pela seguinte formulação:

(M'):

$$\min \sum_{(i,j) \in A} (c_{ij}x_{ij} + f_{ij}y_{ij}), \quad (2.6)$$

s.a.:

$$\sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} = \begin{cases} - \sum_{k \in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \quad (2.7)$$

$$x_{ij} \leq \left(\sum_{k \in D} d_k \right) y_{ij}, \quad \forall (i, j) \in K, \quad (2.8)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in K, \quad (2.9)$$

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in K, \quad (2.10)$$

$$x_{ij} \leq \sum_{k \in D} d_k, \quad \forall (i, j) \in K_1, \quad (2.11)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in K_1, \quad (2.12)$$

$$y_{ij} = 1, \quad \forall (i, j) \in K_1, \quad (2.13)$$

$$x_{ij} = 0, \quad \forall (i, j) \in K_0, \quad (2.14)$$

$$y_{ij} = 0, \quad \forall (i, j) \in K_0. \quad (2.15)$$

A formulação (M') é mais conveniente aos nossos propósitos. Primeiro, porque é imediato que esse modelo representa exatamente o mesmo problema que o modelo (M), se considerarmos $K_1 = K_0 = \emptyset$. Além disso, ao contrário de (M), (M') consegue representar cada um dos subproblemas gerados pelo algoritmo *branch-and-bound*, na árvore de busca.

```

algorithm Solve( $M'$ )
    /* bounding */
    Compute_Lower_and_Upper_Bounds( $L, U$ )
    Update_Best_Upper_Bound( $U, U_{\text{BEST}}$ )
    GAP  $\leftarrow \frac{U-L}{U}$ 
    /* branching */
    if  $L > U_{\text{BEST}}$  then
        write 'Infeasible node reached.'
    else if GAP  $\leq \varepsilon$  then
        write 'Optimum reached.'
    else if  $K = \emptyset$  then
        write 'Leaf reached.'
    else
        Reduce( $M'$ )
         $(i, j) \leftarrow \text{Chosen\_Arc\_}$ 
         $K \leftarrow K \setminus (i, j); K_1 \leftarrow K_1 \cup (i, j)$ 
        Solve( $M'$ )
         $K_1 \leftarrow K_1 \setminus (i, j); K_0 \leftarrow K_0 \cup (i, j)$ 
        Solve( $M'$ )
         $K_0 \leftarrow K_0 \setminus (i, j); K \leftarrow K \cup (i, j)$ 
        Unreduce( $M'$ )
    end if
end if
end if
end algorithm

```

Figura 2.2: Algoritmo *Branch-and-Bound* para o Problema NCFCF

Finalmente, (M') é capaz de descrever o problema NCFCF, mesmo após a aplicação dos testes de redução.

2.3 Método de Solução

Conforme já falado anteriormente, precisamos empregar um esquema enumerativo, se o objetivo for resolver exatamente esse problema \mathcal{NP} -árduo. Assim, usamos o algoritmo *branch-and-bound* apresentado na Figura 2.2. A estratégia de busca escolhida é a em profundidade (*depth-first search*), por medida de economia de memória. A versão utilizada é recursiva, por ser mais fácil de entender e implementar. O algoritmo começa entrando em um processo iterativo para calcular os limites inferior e superior, conforme explicitado mais à frente. Se ao final das iterações os melhores limites L e U forem próximos o bastante, o problema foi resolvido. Caso contrário, o problema sofre uma redução, segundo um algoritmo definido adiante, uma variável é escolhida (conhecida como variável de *branch-*

ing), um subproblema é criado pela sua fixação em um dos valores possíveis e o algoritmo recomeça do início, com novo cálculo dos limites.

O algoritmo na Figura 2.2 pode resolver o problema, uma vez que enumera implicitamente todas as possíveis soluções, mas é computacionalmente ineficiente devido à sua complexidade de pior caso ser exponencial com o tamanho da entrada, $O(2^{|K|})$. O algoritmo é aceitável apenas para pequenas instâncias. Entretanto, bons limites aliados a estratégias eficientes de *branching* e boas técnicas de redução podem aumentar consideravelmente o tamanho das instâncias tratáveis, como demonstraremos mais à frente.

2.3.1 Cálculo dos Limites L e U

Cálculo dos Limites Inferiores L

O uso da relaxação de programação linear (PL) é uma maneira tradicional para calcular limites inferiores para problemas lineares inteiros. Entretanto, usamos a técnica da relaxação lagrangeana. Conforme observado anteriormente, (Fisher, 1985), o uso dessa técnica em problemas como o NCFCF não garante limites inferiores mais justos do que aqueles que seriam obtidos pela relaxação de PL, pois a relaxação lagrangeana utilizada atende à propriedade da integralidade, o que confirmamos no Capítulo 5. No entanto, como mostraremos a seguir, será possível derivar para o problema uma heurística baseada na relaxação lagrangeana, com limites superiores bastante justos. Além do mais, a técnica tem sido aplicada extensivamente a importantes casos particulares do problema NCFCF. A relaxação lagrangeana tem sido usada com sucesso na resolução de problemas de planejamento de sistemas de distribuição, (Rothfarb *et al.*, 1970), problemas de localização capacitada, (Geoffrion e McBride, 1978), localização não-capacitada, (Galvão e Raggi, 1989, Mateus e Carvalho, 1992), e modelos orientados a aplicações em redes de telecomunicações, (Gavish, 1991, Gavish, 1992).

Relaxando as restrições de capacidade (2.8) por meio das variáveis duais $w_{ij} \geq 0$, $\forall (i, j) \in K$, a seguinte função lagrangeana resulta:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{(i,j) \in A} (c_{ij}x_{ij} + f_{ij}y_{ij}) + \sum_{(i,j) \in K} w_{ij} \left[x_{ij} - \left(\sum_{k \in D} d_k \right) y_{ij} \right]. \quad (2.16)$$

Consequentemente, uma relaxação lagrangeana para (M') pode ser escrita como segue: $(LR_{\mathbf{w}})$:

$$L(\mathbf{w}) = \min\{\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{w}) \text{ s.a.: (2.7), (2.9), (2.10), (2.11), (2.12), (2.13), (2.14), (2.15), } \mathbf{w} \geq 0\}. \quad (2.17)$$

Para qualquer vetor viável de multiplicadores de Lagrange, $\mathbf{w} \geq \mathbf{0}$, a solução ótima da relaxação lagrangeana é um limite inferior para o problema original, pois a quantidade $\sum_{(i,j) \in K} w_{ij} [x_{ij}^* - (\sum_{k \in D} d_k) y_{ij}^*]$ é sempre não-positiva, supondo que $L(\mathbf{w}) = \mathcal{L}(\mathbf{x}^*, \mathbf{y}^*; \mathbf{w})$, (Fisher, 1985). Portanto, o cálculo dos limites inferiores é reduzido à resolução de dois

subproblemas fáceis (polimoniais), (i) um problema linear de fluxos em \mathbf{x} e (ii) um problema de seleção em \mathbf{y} , a saber:

(LR₁):

$$\min \sum_{(i,j) \in A} C_{ij} x_{ij}, \quad (2.18)$$

s.a.:

$$\sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} = \begin{cases} - \sum_{k \in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \quad (2.19)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in K, \quad (2.20)$$

$$x_{ij} \leq \sum_{k \in D} d_k, \quad \forall (i, j) \in K_1, \quad (2.21)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in K_1, \quad (2.22)$$

$$x_{ij} = 0, \quad \forall (i, j) \in K_0, \quad (2.23)$$

onde

$$C_{ij} = \begin{cases} c_{ij} + w_{ij}, & \forall (i, j) \in K, \\ c_{ij}, & \forall (i, j) \in A \setminus K. \end{cases} \quad (2.24)$$

(LR₂):

$$\min \sum_{(i,j) \in A} F_{ij} y_{ij}, \quad (2.25)$$

s.a.:

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in K, \quad (2.26)$$

$$y_{ij} = 1, \quad \forall (i, j) \in K_1, \quad (2.27)$$

$$y_{ij} = 0, \quad \forall (i, j) \in K_0, \quad (2.28)$$

onde

$$F_{ij} = \begin{cases} f_{ij} - w_{ij} \sum_{k \in D} d_k, & \forall (i, j) \in K, \\ f_{ij}, & \forall (i, j) \in A \setminus K. \end{cases} \quad (2.29)$$

O problema em \mathbf{x} pode ser resolvido com complexidade de tempo polinomial $O(|N||A|)$, (Bazaraa *et al.*, 1990), através do algoritmo de caminhos mínimos para custos arbitrários, (Glover *et al.*, 1985), lembrando que, segundo nossa hipótese inicial, não há circuitos com custos negativos¹. O problema em \mathbf{y} pode ser resolvido com complexidade de tempo polinomial, $O(|A|)$, pois tudo que precisamos é fazer $y_{ij} = 1$, para todo $F_{ij} < 0$.

¹Note que o problema de caminhos mínimos, sem ciclos, na presença de circuitos negativos é \mathcal{NP} -árduo, (Bazaraa *et al.*, 1990).

```

procedure Compute_Lower_and_Upper_Bounds( $L, U$ )
   $\mathbf{w}_0 \leftarrow \mathbf{0}$ 
  initialize lower bound  $L$ ,  $L \leftarrow -\infty$ 
  initialize upper bound  $U$ ,  $U \leftarrow +\infty$ 
   $k \leftarrow 0$ 
  repeat
    compute current lower bound  $L_{\mathbf{w}_k}$  and update  $L$ 
    compute current upper bound  $U_{\mathbf{w}_k}$  and update  $U$ 
     $\text{GAP} \leftarrow \frac{U-L}{U}$ 
    compute subgradient vector  $\boldsymbol{\gamma}_{\mathbf{w}_k}$ 
    if ( $\text{GAP} > \varepsilon$ ) and ( $\|\boldsymbol{\gamma}_{\mathbf{w}_k}\|^2 \neq 0$ ) then
      compute step  $t_k$ 
       $\mathbf{w}_{k+1} \leftarrow \max\{0, (\mathbf{w}_k + t_k * \boldsymbol{\gamma}_{\mathbf{w}_k})\}$ 
       $k \leftarrow k + 1$ 
    end if
  until ( $\text{GAP} \leq \varepsilon$ ) or ( $\|\boldsymbol{\gamma}_{\mathbf{w}_{k-1}}\|^2 = 0$ ) or ( $t_{k-1} \leq \varepsilon$ )
  return  $L, U$ 
end procedure

```

Figura 2.3: Algoritmo de Otimização por Subgradientes

Cálculo dos Limites Superiores U

Propomos uma maneira simples e eficiente para calcular os limites superiores. Os fluxos obtidos pela resolução do problema (LR_1) são soluções viáveis do problema primal, uma vez que atendem às demandas. O único trabalho adicional é calcular o custo total dos fluxos, usando os custos c_{ij} originais, e adicionar a esse valor o custo fixo total associado aos arcos utilizados, empregando os custos fixos f_{ij} também originais.

Melhoria dos Limites L e U

O modo tradicional de melhorar os limites inferiores e superiores obtidos via relaxação lagrangeana é aplicar um algoritmo de otimização por subgradientes, (Held *et al.*, 1974), Figura 2.3. Nesse trabalho, inicializamos os multiplicadores de Lagrange \mathbf{w} por $\mathbf{0}$ somente na primeira chamada ao algoritmo. Nas chamadas subsequentes, não há necessidade, pois não há razão por que o vetor usado pela última vez não possa ser utilizado novamente como o vetor \mathbf{w}_0 de ponto de partida.

Supondo que $L(\mathbf{w}) = \mathcal{L}(\mathbf{x}^*, \mathbf{y}^*; \mathbf{w})$, ver a Equação (2.17), um vetor subgradiente da função L , no ponto \mathbf{w} , é dado por:

$$\boldsymbol{\gamma}_{\mathbf{w}} = \left\{ \left[\begin{array}{c} x_{ij}^* - \left(\sum_{k \in D} d_k \right) y_{ij}^* \\ \end{array} \right]_{(i,j) \in K} \right\}. \quad (2.30)$$


```

procedure Chosen_Arc_First_Free
  for all  $(i, j) \in A$  do
    if  $(i, j) \in K$  then
      return  $(i, j)$ 
    end if
  end for
  return FAIL
end procedure

```

Figura 2.4: Estratégia Ingênua de *Branching* para o Problema NCFCF

A definição do tamanho do passo t_k apresentada abaixo tem se mostrado eficaz na prática, (Held *et al.*, 1974, Christofides e Beasley, 1982, Fisher, 1985, Hochbaum e Segev, 1989, Beasley, 1989):

$$t_k = C_k \frac{U - L_{\mathbf{w}_k}}{\|\gamma_{\mathbf{w}_k}\|^2}. \quad (2.31)$$

Os valores usados para o coeficiente de dilatação C_k são cruciais para o bom funcionamento do algoritmo. Embora existam estudos mais recentes no que se refere à escolha desses valores, (Santos e Galvão, 1992), decidimos usar um esquema tradicional, (Held *et al.*, 1974). O valor de C_k é mantido igual a 2 por $2|A|$ iterações, sendo $|A|$ a cardinalidade do conjunto de arcos, portanto, o tamanho da entrada. C_k é reduzido à metade, assim como o número de iterações no qual C_k é mantido fixo. Quando esse número atinge o limiar de 5, C_k é dividido por 2 a cada 5 iterações, até que o tamanho do passo t_k caia abaixo do valor ε pré-estabelecido.

2.3.2 Estratégias de *Branching*

A estratégia de escolha da variável de *branching* é fundamental no desempenho de algoritmos *branch-and-bound*. O uso de escolhas inteligentes, associadas a bons limitantes, pode vir a reduzir significativamente o número de nós explorados na árvore de pesquisa, com consequente redução no tempo de processamento. Usaremos duas estratégias diferentes.

O método mais simples de escolha é retornar a primeira variável livre encontrada, Figura 2.4. O procedimento tem uma complexidade de tempo $O(|A|)$, no pior caso, uma vez que o teste “**if** $(i, j) \in K$ **then**” pode ser feito $O(1)$. Entretanto, as soluções ótimas do problema NCFCF, formulado conforme o modelo (M) , possui uma importante propriedade que pode melhorar consideravelmente o desempenho do algoritmo *branch-and-bound*. A propriedade é apresentada pelo Teorema 2.1, (Cruz *et al.*, 1996b), abaixo:

Teorema 2.1 *Se o modelo (M) possuir um ótimo finito, então haverá um conjunto ótimo de arcos com fluxos positivos de tal forma que, no máximo, apenas um arco incida em cada nó.*

```

procedure Chosen_Arc_No_Cycling
  /* compute set of reached nodes  $E$  */
   $E \leftarrow \{s\}$ 
  for all  $(i, j) \in A$  do
    if  $(i, j) \in K_1$  then
       $E \leftarrow E \cup \{i\} \cup \{j\}$ 
    end if
  end for
  /* search new eligible free arc */
  for all  $(i, j) \in A$  do
    if  $(i, j) \in K$  and  $i \in E$  and  $j \notin E$  then
      return  $(i, j)$ 
    end if
  end for
  return FAIL
end procedure

```

Figura 2.5: Estratégia Melhorada de *Branching* para o Problema NCFCF

Prova (por contradição): *Suponhamos que o Teorema 2.1 não seja satisfeito por nenhuma solução ótima e que o nó m possua dois arcos incidentes, (k, m) e (l, m) . Deverá existir um conjunto de arcos formando um caminho sem ciclos da fonte s até o nó m passando pelo nó k , i.e. usando o arco (k, m) , chamado P_k . Similarmente, deverá haver um caminho usando o arco (l, m) , chamado P_l . Sem perda de generalidade, suponhamos também que*

$$\sum_{(i,j) \in P_k} c_{ij} \leq \sum_{(i,j) \in P_l} c_{ij}.$$

Assim, desabilitando o arco (l, m) e transferindo o fluxo x_{lm} do caminho P_l para o caminho P_k , haverá, no mínimo, a seguinte redução na função objetivo:

$$\left(\sum_{(i,j) \in P_l} c_{ij} - \sum_{(i,j) \in P_k} c_{ij} \right) x_{ij} + f_{lm} \geq 0.$$

A solução resultante é, pelo menos, tão boa quanto a inicial e satisfaz ao Teorema 2.1, contrariando a nossa suposição inicial. ■

Assim, uma outra possibilidade é escolher uma variável livre que não viole o Teorema 2.1, Figura 2.5, (Cruz *et al.*, 1996b). O último comando “**for all** $(i, j) \in A$ **do comandos end for**”, no procedimento da Figura 2.5, é a operação mais demorada, devendo terminar após $O(|A|)$ iterações, no pior caso. Seu teste interno “**if** $(i, j) \in K$ **and** $i \in E$ **and** $j \notin E$ **then**” pode ser feito $O(1)$, resultando em uma complexidade global de pior caso $O(|A|)$, surpreendentemente a mesma do procedimento da Figura 2.4. Assim, de antemão, é possível afirmar com certeza que essa última estratégia, no pior caso, terá um desempenho pelo menos tão bom quanto o da estratégia anterior.

```

procedure Reduce( $M'$ )
    /* initialize set of arcs recently fixed */
     $F \leftarrow \emptyset$ 
    /* proceed with reduction */
    for all  $(i, j) \in A$  do
        if  $(i, j) \in K$  then
             $K \leftarrow K \setminus (i, j)$ 
             $K_1 \leftarrow K_1 \cup (i, j)$ 
            if  $L(\mathbf{w}) > U_{\text{BEST}}$  then
                 $K_1 \leftarrow K_1 \setminus (i, j)$ 
                 $K_0 \leftarrow K_0 \cup (i, j)$ 
                 $F \leftarrow F \cup (i, j)$ 
            else
                 $K_1 \leftarrow K_1 \setminus (i, j)$ 
                 $K_0 \leftarrow K_0 \cup (i, j)$ 
                if  $L(\mathbf{w}) > U_{\text{BEST}}$  then
                     $K_0 \leftarrow K_0 \setminus (i, j)$ 
                     $K_1 \leftarrow K_1 \cup (i, j)$ 
                     $F \leftarrow F \cup (i, j)$ 
                else
                     $K_0 \leftarrow K_0 \setminus (i, j)$ 
                     $K \leftarrow K \cup (i, j)$ 
                end if
            end if
        end if
    end for
end procedure

```

Figura 2.6: Teste de Redução para o Problema NCFCF

2.3.3 Redução do Problema

O novo limite inferior que resultaria de forçarmos o arco (i, j) a estar ou não na solução pode ser facilmente estimado pela relaxação lagrangeana. Se o limite inferior resultante da imposição de alguma condição à relaxação lagrangeana é maior do que o melhor limite superior até então obtido, então a condição em consideração não poderia ser satisfeita no ótimo. A idéia do método é inspirada em procedimentos de redução desenvolvidos para o problema das p -medianas, (Christofides e Beasley, 1982). Naquele caso, alguns termos da relaxação lagrangeana correspondente foram usados para estimar o incremento no limite inferior, causado pela condição imposta. Propomos o uso do procedimento apresentado na Figura 2.6, (Cruz *et al.*, 1996d), que usa limites inferiores estimados pela resolução completa da relaxação lagrangeana $L(\mathbf{w})$, mas sem iterações de subgradiente. Definimos F como o conjunto de todos os arcos que foram fixados pelo procedimento de redução.

```

procedure Unreduce( $M'$ )
  for all  $(i, j) \in A$  do
    if  $(i, j) \in F$  then
       $K \leftarrow K \cup (i, j)$ 
      if  $(i, j) \in K_0$  then
         $K_0 \leftarrow K_0 \setminus (i, j)$ 
      else
         $K_1 \leftarrow K_1 \setminus (i, j)$ 
      end if
    end if
  end for
end procedure

```

Figura 2.7: Procedimento de Restabelecimento para o Problema NCFCF

O procedimento de redução é finalizado após o exame de cada arco exatamente uma vez, o que é $O(|A|)$. Algumas variantes para o algoritmo são imediatas, *e.g.* passar duas vezes pelos arcos, *etc.* Cada uma dessas iterações envolve no máximo dois cálculos de limite inferior, que são $O(|N||A|)$ cada. Lembramos que aqui não há aplicação do algoritmo de otimização por subgradientes. Além disso, no máximo quatro inserções e retiradas em conjuntos estão envolvidas, as quais são $O(|A|)$, mas o cálculo da função $L(\mathbf{w})$ é dominante. Portanto, o procedimento terá uma complexidade polinomial $O(|N||A|^2)$, no pior caso.

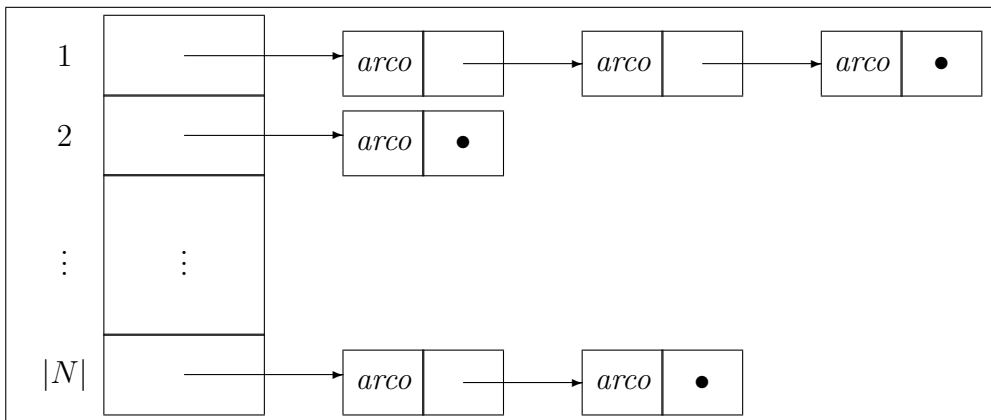
Como observação final, o procedimento $O(|A|^2)$ da Figura 2.7 deverá ser ativado, no estágio de *back-tracking*, para restabelecer o problema (M'), devolvendo-o ao estado anterior à aplicação do algoritmo de redução, (Cruz *et al.*, 1996d).

2.3.4 Projeto das Estruturas de Dados

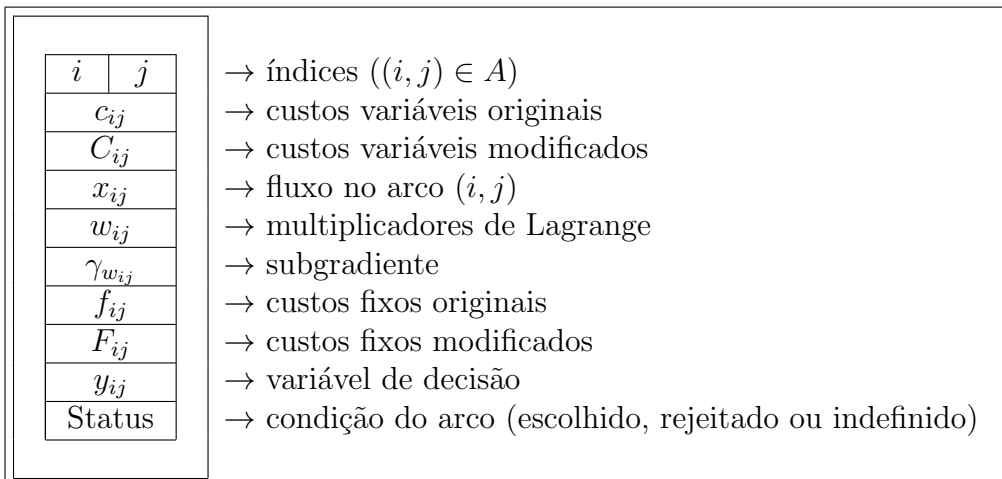
As estruturas de dados desempenham um papel importante no funcionamento dos algoritmos e nos requisitos de armazenagem. Todas as complexidades computacionais apresentadas até esse ponto são fortemente dependentes das estruturas de dados empregadas.

Um modo eficiente de representar grafos é por meio de uma lista de adjacências, Figura 2.8. A representação de conjuntos também é necessária, *e.g.* conjuntos D , E e F . Uma escolha bastante aceitável é por meio de listas encadeadas, Figura 2.9. Entretanto, quanto ao conjunto de nós alcançados E , Figura 2.5, a melhor representação é por meio de um vetor com $|N|$ posições. Essa não é a forma mais econômica, mas será usada por ser capaz de suportar operações críticas, tais como “ $i \in E?$ ” e “ $E \leftarrow E \cup \{i\}$ ”, com complexidade de tempo constante $O(1)$. Essas estruturas de dados tornam possível uma implementação rápida e elegante, independente do tamanho da entrada.

Além disso, a representação por listas de adjacências permite economia de espaço, na medida em que os dígrafos armazenados forem esparsos, (Aho *et al.*, 1983, Ziviani, 1993). A complexidade de cada iteração do algoritmo de otimização por subgradientes



a) Lista de Adjacência



b) Definição do Elemento *arco*

Figura 2.8: Representação do Dígrafo $\mathcal{D} = (N, A)$ por Listas de Adjacências

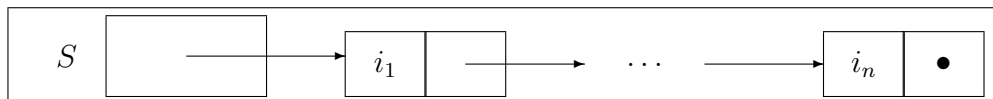


Figura 2.9: Representação do Conjunto S por Listas Encadeadas

será $O(|N||A|)$, que é a complexidade do algoritmo de rotulação empregado para resolver o problema linear de fluxos em \mathbf{x} . Essa complexidade só é garantida com a representação de dígrafos por listas de adjacências, (Glover *et al.*, 1985, Bazaraa *et al.*, 1990).

2.4 Resultados Experimentais

Todos os resultados apresentados foram executados em uma DECstation 3100, com o sistema operacional ULTRIX V4.2A, usando uma versão preliminar dos algoritmos codificada na linguagem C. O parâmetro ε usado no algoritmo *branch-and-bound* e no algoritmo de otimização por subgradientes é 10^{-6} , que é próximo à precisão oferecida pelo compilador empregado, quando representa números reais pelo tipo *float*.

Os problemas de teste vieram de grafos euclidianos gerados aleatoriamente, por meio de um esquema conhecido, (Aneja, 1980) e extensivamente usado para testar algoritmos para o problema de *Steiner* em grafos, (Wong, 1984, Beasley, 1989, Duin e Volgenant, 1989b). Os pesos Ω_{ij} nos arcos foram definidos como sendo a distância euclidiana entre as suas extremidades i e j . Os problemas apresentados na Tabela 2.1 foram gerados por um programa em C que desenvolvemos. Os problemas da Tabela 2.2 fazem parte de uma biblioteca de problemas de teste, (Beasley, 1990), já usados para testar algoritmos para o problema de *Steiner* em grafos, (Beasley, 1989, Duin e Volgenant, 1989b).

As demandas foram consideradas unitárias. Os custos f_{ij} e c_{ij} foram derivados dos pesos Ω_{ij} , usando os fatores 1 e 10. Para cada grafo, três diferentes instâncias com diferentes razões f_{ij}/c_{ij} foram consideradas. Os problemas usando a razão 1 : 10 formam uma classe de problemas “próximos” ao problema linear de fluxos, que é polinomial. Os problemas com razão 10 : 1 formam uma classe de problemas “próximos” ao problema de *Steiner* em grafos, que é \mathcal{NP} -árduo. Entretanto, ambas as classes mencionadas continuam \mathcal{NP} -árduas.

Para cada caso de teste, apresentamos os seguinte resultados para o nó inicial da árvore de busca: o melhor limite superior, o *gap*, o tempo de UCP, em segundos, excluindo toda operação de entrada e saída e considerando que somente um processo estava rodando na máquina. Para o restante da árvore de busca, a menos que obtivéssemos um *overflow* de tempo de 20.000 segundos, apresentamos a solução ótima, o número de nós e o tempo de UCP, em segundos, gasto para finalização de cada uma das quatro combinações seguintes: (i) usando a estratégia de *branching* da Figura 2.4, (ii) usando a estratégia melhorada de *branching*, Figura 2.5, (iii) combinando essa última com o algoritmo de redução da Figura 2.6, e (iv) usando essa terceira combinação, mas excluindo as iterações de subgradiente após a exploração do primeiro nó da árvore de pesquisa.

Dos resultados apresentados nas Tabelas 2.1 e 2.2, pode ser visto que, embora oferecendo limites inferiores relativamente fracos, principalmente para os problemas mais próximos aos problemas de *Steiner*, a relaxação lagrangeana aqui desenvolvida é uma boa heurística para resolução dos problemas NCFCF. Somente dois dos problemas das Tabelas 2.1 e 2.2, entre aqueles resolvidos até a otimalidade, é que esse ótimo não foi alcançado no primeiro nó da árvore de busca. Claro que quanto maiores e mais densos os problemas forem, menor será a chance de que tal fato ocorra.

Tabela 2.1: Resultados Computacionais para Redes Aleatórias, com $|N| = 16$ e $|N| = 32$

$ N $	$ A $	$ D $	$\frac{f_{ij}}{\Omega_{ij}}$	$\frac{c_{ij}}{\Omega_{ij}}$	Árvore de Busca												
					Primeiro Nó			<i>Chosen_Arc_First_Free</i>		<i>Chosen_Arc_No_Cycling</i>		Reduzido		Simplificado			
					U_{BEST}	GAP(%)	UCP(s)	U_{OPT}	Nós	UCP(s)	Nós	UCP(s)	Nós	UCP(s)	Nós	UCP(s)	
16	30	4	1	10	5.972	1,50	0,20	5.972	301	23,00	35	2,80	1	0,03	1	0,03	
			1	1	806	11,00	0,20	806	301	23,00	35	2,70	1	0,03	1	0,03	
			10	1	2.894	31,00	0,22	2.894	301	25,00	35	3,00	1	0,03	1	0,03	
		8	1	10	10	12.250	2,10	0,21	12.250	819	68,00	35	2,90	1	0,04	1	0,04
				1	1	1.585	16,00	0,21	1.585	819	68,00	35	2,90	1	0,04	1	0,04
				10	1	5.185	49,00	0,22	5.185	819	73,00	35	3,20	1	0,04	1	0,04
		60	4	1	10	4.066	4,20	0,52	4.066	9.445	1.900,00	85	19,00	1	0,12	1	0,12
				1	1	646	26,00	0,53	646	18.941	3.700,00	259	61,00	31	16,00	57	5,10
				10	1	2.400	45,00	0,59	2.400	**	**	557	150,00	43	23,00	57	5,50
32	62	4	1	10	7.645	4,00	0,64	7.645	**	**	905	240,00	1	0,18	1	0,18	
			1	1	1.201	25,00	0,65	1.201	**	**	905	230,00	1	0,18	1	0,18	
			10	1	5.566	55,00	0,72	5.566	**	**	905	250,00	1	0,17	1	0,18	
		8	1	10	12.349	3,60	0,66	12.349	**	**	4.261	1.100,00	1	0,19	1	0,20	
			1	1	1.765	25,00	0,64	1.765	**	**	4.261	1.100,00	1	0,19	1	0,20	
			10	1	7.066	63,00	0,67	7.066	**	**	4.261	1.100,00	1	0,20	1	0,20	
	16	1	10	30.093	2,60	0,69	30.093	**	**	2.255	640,00	1	0,21	1	0,21		
		1	1	3.885	20,00	0,69	3.885	**	**	2.255	641,00	1	0,21	1	0,21		
		10	1	12.642	63,00	0,73	12.642	**	**	2.255	670,00	1	0,21	1	0,21		
	31	1	10	62.499	2,20	0,80	62.499	**	**	61	22,00	1	0,24	1	0,24		
		1	1	7.644	18,00	0,77	7.644	**	**	65	22,00	1	0,24	1	0,25		
		10	1	21.585	63,00	0,82	21.585	**	**	65	23,00	1	0,25	1	0,25		
	124	4	1	10	6.891	3,20	1,90	6.891	**	**	2.255	2.400,00	9	18,00	9	4,00	
			1	1	1.026	21,00	1,90	1.016	**	**	**	**	165	340,00	269	110,00	
			10	1	3.878	41,00	2,30	3.878	**	**	**	**	597	1.200,00	969	360,00	
		8	1	10	14.484	3,00	2,00	14.484	**	**	**	**	15	35,00	15	9,30	
			1	1	1.986	21,00	2,00	**	**	**	**	**	**	**	**	**	
			10	1	6.845	54,00	2,10	**	**	**	**	**	**	**	**	**	
16		1	10	25.023	2,30	2,10	25.023	**	**	**	**	19	46,00	19	13,00		
		1	1	3.147	18,00	2,70	**	**	**	**	**	**	**	**	**		
		10	1	8.964	56,00	2,20	**	**	**	**	**	**	**	**	**		
248		4	1	10	2.821	5,90	6,40	2.821	**	**	**	**	63	420,00	63	83,00	
			1	1	468	31,00	6,40	468	**	**	**	**	337	2.100,00	367	360,00	
			10	1	1.926	53,00	6,80	**	**	**	**	**	**	**	**	**	
	8	1	10	5.200	6,80	6,40	5.200	**	**	**	**	679	4.900,00	679	1.100,00		
		1	1	861	37,00	6,60	**	**	**	**	**	**	**	**	**		
		10	1	3.696	71,00	6,60	**	**	**	**	**	**	**	**	**		

**Não disponível (*overflow* de tempo)

Tabela 2.2: Resultados Computacionais para as Redes de Beasley, (Beasley, 1990), com $f_{ij}/\Omega_{ij} = 1$ e $c_{ij}/\Omega_{ij} = 10$

Problema	$ N $	$ A $	$ D $	Árvore de Busca													
				Primeiro Nó			<i>Chosen_Arc_First_Free</i>		<i>Chosen_Arc_No_Cycling</i>		Reduzido		Simplificado				
				U_{BEST}	GAP(%)	UCP(s)	U_{OPT}	Nós	UCP(s)	Nós	UCP(s)	Nós	UCP(s)	Nós	UCP(s)		
B-1	50	126	8	1.222	5,60	2,30	1.222	**	**	**	**	1	1,10	1	1,10		
2			12	2.520	2,80	2,30	2.520	**	**	**	**	1	3,20	1	2,70		
3			24	5.017	3,10	2,40	5.012	**	**	**	**	495	1.300,00	281	170,00		
4		200	8	1.237	5,10	4,90	**	**	**	**	**	**	**	**	**		
5			12	1.095	5,20	4,80	1.095	**	**	**	**	1.676	8.800,00	1.799	1.800,00		
6			24	3.208	4,20	5,90	**	**	**	**	**	**	**	**	**		
7	75	188	12	2.943	3,40	4,80	2.943	**	**	**	**	1	4,40	1	4,50		
8			18	2.657	3,90	5,00	2.657	**	**	**	**	9	55,00	7	16,00		
9			37	5.874	3,70	5,20	5.874	**	**	**	**	5	31,00	5	12,00		
10		300	12	2.053	5,20	11,00	**	**	**	**	**	**	**	**	**		
11			18	3.987	2,70	11,00	3.987	**	**	**	**	67	1.100,00	61	297,00		
12			37	6.948	3,00	12,00	**	**	**	**	**	**	**	**	**		
13	100	250	16	4.432	3,70	9,30	4.432	**	**	**	**	**	**	6.517	19.000,00		
14			24	9.117	2,60	9,50	**	**	**	**	**	**	**	**	**		
15			49	11.383	2,90	9,40	11.383	**	**	**	**	**	**	1.923	4.800,00		
16		400	16	3.942	3,50	24,00	**	**	**	**	**	**	**	**	**		
17			24	5.193	2,40	24,00	**	**	**	**	**	**	**	**	**		
18			49	6.360	4,20	24,00	**	**	**	**	**	**	**	**	**		

**Não disponível (*overflow* de tempo)

Os resultados também mostram o efeito no tempo de processamento da estratégia melhorada de *branching*. A redução no tempo de processamento é causada pela redução no número de combinações, uma vez que o procedimento evita todas aquelas que conduzam a ciclos. O impacto do algoritmo de redução também é notável. Nos problemas esparsos, o algoritmo manteve o número de nós explorados surpreendentemente baixo. O algoritmo foi capaz de resolver rapidamente problemas densos, com baixo número de nós de demanda e mais próximos ao problema linear de fluxos. Os problemas próximos ao problema de *Steiner* foram realmente mais difíceis.

Uma interessante questão emersa durante o desenvolvimento desse trabalho foi o efeito do uso subsequente de iterações de subgradiente. Notamos, na prática, que o algoritmo de otimização por subgradientes oferecia apenas pequenas melhorias nos limites inferiores, após o primeiro nó da árvore de busca. Após a redução e fixação das variáveis de decisão, os multiplicadores remanescentes continuam viáveis e tais valores já são muito próximos dos ótimos. Conseqüentemente, os limites inferiores que podem ser obtidos por esses multiplicadores deverão ser também muito bons. Assim, decidimos aplicar o algoritmo de otimização por subgradientes apenas uma vez. Os resultados são mostrados nas duas últimas colunas das Tabelas 2.1 e 2.2. Em algumas instâncias, foi possível observar um pequeno incremento no número de nós explorados. Entretanto, em todos os casos testados, o decréscimo no tempo de processamento foi efetivo.

2.5 Conclusões

A obtenção de soluções exatas para o problema NCFCF não é trivial, uma vez que se trata de um problema de otimização \mathcal{NP} -árduo. Algoritmos *branch-and-bound* são a possibilidade mais preferida e frequentemente são aceitáveis apenas para pequenas instâncias, devido ao seu tempo de processamento explosivo. Propomos um conjunto de procedimentos auxiliares que reduzem os tempos de processamento, conseqüentemente tornando possível o tratamento de instâncias maiores. Por meio de experimentos, mostramos que o nosso critério de *branching* e a nossa técnica de redução podem realmente acelerar o processo de obtenção de uma solução ótima. Adicionalmente, as idéias por trás desses procedimentos são aplicáveis a problemas similares. De fato, estendemos os algoritmos aqui apresentados ao problema de planejamento de redes em multi-níveis, (Cruz *et al.*, 1996c), conforme apresentamos no Capítulo 3.

Algumas questões permanecem abertas. Seria possível desenvolver alguma estratégia para escolha de arcos, entre todos os livres que não formam ciclos, em vez de escolher o primeiro deles, como fazemos aqui? O teste de redução poderia ser mais eficiente se cada arco fosse examinado mais de uma vez? Trabalhos futuros poderiam incluir a investigação de tais questões. Poderiam também incluir o desenvolvimento de testes de redução adicionais, como aqueles que eliminam nós e contraem arcos, similares aos já conhecidos para o problema de *Steiner* em grafos, (Duin e Volgenant, 1989b, Maculan *et al.*, 1991). Seria também interessante estudar como as idéias aqui desenvolvidas poderiam ser adaptadas aos eficientes algoritmos especializados nos casos particulares

do problema NCFCF, *e.g.* o problema de transporte com custos fixos, (Barr *et al.*, 1981), o problema de *Steiner* em grafos, (Beasley, 1989), ou ainda o problema de localização não-capacitado, (Galvão, 1993).

Capítulo 3

O Problema de Planejamento de Redes em Multi-Níveis

Nesse Capítulo, introduzimos o problema de planejamento de redes em multi-níveis (PRMN), que é um modelo integrando três importantes problemas de otimização em redes: localização discreta de facilidades, planejamento topológico e dimensionamento de redes. Discutimos aplicações potenciais para o modelo. Definimos matematicamente o problema e apresentamos duas formulações equivalentes. Propomos um algoritmo *branch-and-bound* e um novo conjunto de poderosos procedimentos auxiliares. A experiência computacional que apresentamos atestam a qualidade dos nossos resultados, (Cruz *et al.*, 1996e).

3.1 Introdução

O problema PRMN integra aspectos de localização discreta de facilidades, planejamento topológico de redes e dimensionamento. Uma rede em multi-níveis é ilustrada no Capítulo 1, Figura 1.1. O problema PRMN é definido em um dígrafo multi-ponderado $\mathcal{D} = (N, A)$, onde N é o conjunto de nós e A , o conjunto de arcos. Os candidatos a nós de oferta são agrupados em m conjuntos, de acordo com a sua habilidade de produzir um determinado tipo de fluxo, onde m é o número de níveis presentes. Os nós de demanda também são agrupados em m conjuntos. Nós de transbordo ou de *Steiner* podem ou não estar presentes. O objetivo é determinar uma combinação ótima de nós de oferta e arcos, de forma a levar o tipo de fluxo requerido a cada um dos nós de demanda, respeitando certas regras de conservação e transformação de fluxos. Os candidatos a nós de oferta não podem criar fluxos, exceto aqueles do primeiro nível, mas podem receber um fluxo do l -ésimo tipo e transformá-lo em fluxo do $(l + 1)$ -ésimo tipo, usando a razão 1 : 1. Os custos envolvidos são os custos fixos associados à escolha dos candidatos a nós de oferta, os custos fixos dos arcos escolhidos e os custos variáveis dos fluxos.

O estudo de redes em multi-níveis é importante em termos teóricos porque elas podem ser vistas como generalização de importantes problemas de otimização em redes, tais como problemas de planejamento topológico, problemas de fluxos com custos fixos ou mesmo

problemas de localização não-capacitada. Uma extensa e compreensível bibliografia sobre problemas de planejamento topológico de redes pode ser encontrada *e.g.* em (Gavish, 1982). Muitos problemas de otimização em redes aplicados às telecomunicações têm sido estudados, incluindo problemas de planejamento de topologia de redes, (Gavish, 1992, Balakrishnan *et al.*, 1994a), problemas de topologia integrados aos problemas de dimensionamento, (Luna *et al.*, 1987), e problemas de roteamento, (Balakrishnan e Altinkemer, 1992). Os problemas de planejamento de redes com custos fixos nos arcos, (Nemhauser e Wolsey, 1988), que são um caso particular representando uma importante classe de problemas de programação inteira-mista, também receberam atenção, (Hochbaum e Segev, 1989, Rardin e Wolsey, 1993). O problema de *Steiner* em grafos, (Maculan, 1987), é provavelmente um dos casos particulares mais estudados. Mesmo sendo um modelo clássico, resultados recentes têm sido descobertos para o problema, (Goemans e Myung, 1993, Goemans, 1994). Recentes avanços em algoritmos para o problema de localização não-capacitado, (Erlenkotter, 1978), outro relevante caso particular cuja solução tem muitas implicações no mundo real, também têm sido conseguidos, (Galvão, 1993).

Aplicações potenciais para esse novo modelo incluem planejamento de sistemas elétricos de potência. O modelo proposto consegue representar alguns dos mais importantes problemas técnicos e requisitos de segurança presentes, tais como: (i) a necessidade do uso de linhas de extra-alta tensão (tipicamente de 275 kV a 1.000 kV), para transmissão eficiente da energia, (ii) uso de linhas de tensões intermediárias (11 kV) e baixas (110 e 220 V), em áreas densamente ocupadas, (iii) acomodação das classes de consumidores domésticos, comerciais e industriais, e seus respectivos requisitos de tensão, e (iv) problemas relativos às normas técnicas de segurança para localização de sub-estações.

Redes de telecomunicações são outro bom exemplo de aplicação potencial. Com a introdução da tecnologia digital, novos serviços têm aparecido, agrupando diferentes tipos de fluxos, *e.g.* serviços de voz simultânea com transmissão de imagens e sons digitalizadas. Além disso, diferentes meios de transmissão com diferentes taxas de transmissão deverão coexistir por um bom tempo, *e.g.* rádio-*links*, fibra ótica e cabos de cobre. Acreditamos que o modelo representando o problema PRMN poderia lidar apropriadamente com essa problemática.

Do nosso conhecimento, pouca pesquisa tem sido feita em problemas PRMN. Redes em multi-níveis têm aparecido na literatura recente, mas alguns trabalhos ou não consideram a integração da localização, planejamento e dimensionamento no mesmo modelo, (Current *et al.*, 1986, Balakrishnan *et al.*, 1994a, Balakrishnan *et al.*, 1994b) ou não fornecem uma formulação matemática com definição de limite inferior para o problema, (Cruz, 1991, Cruz *et al.*, 1991, Cruz *et al.*, 1992a, Mateus *et al.*, 1994), como pretendemos fazer aqui.

Como observação final, notamos que de uma maneira geral os problemas em multi-níveis vêm sendo resolvidos por metodologias que adotam uma estratégia que emprega hierarquia de modelos, resolvendo problemas parciais, (Mateus e Luna, 1989, Gavish, 1991, Balakrishnan *et al.*, 1991). Nessa estratégia, o problema geral é decomposto em subproblemas menores de forma a não causar erros apreciáveis. No modelo que estamos introduzindo, não adotamos essa abordagem de modelagem hierárquica.

O problema PRMN é formulado na Seção 3.2. Apresentamos duas formulações ma-

temáticas equivalentes. A Seção 3.3 é dedicada à apresentação de todos os algoritmos desenvolvidos. Entre outras, há uma estratégia para escolha de variável de *branching* e um algoritmo de redução baseado na relaxação lagrangeana. Na Seção 3.4 são apresentados os resultados da nossa experiência computacional. A Seção 3.5 encerra o Capítulo com observações finais e a discussão de possíveis extensões para o trabalho.

3.2 Formulações

A notação abaixo será usada nas formulações matemáticas do problema PRMN:

m - número de níveis;

R^l - conjunto de candidatos a nós de oferta do l -ésimo nível;

D^l - conjunto de nós de demanda do l -ésimo nível;

d_i - demanda do nó $i \in D^l$, por fluxo do l -ésimo nível;

T^l - conjunto de nós de transbordo, definidos como segue: $T^l = N \setminus (R^l \cup D^l \cup R^{l+1})$ para $l = 1, 2, \dots, (m - 1)$, e $T^m = N \setminus (R^m \cup D^m)$;

c_{ij}^l - custo variável (não-negativo) por unidade de fluxo do l -ésimo nível;

x_{ij}^l - fluxo do l -ésimo nível através do arco $(i, j) \in A$;

f_{ij}^l - custo fixo (não-negativo) de ter fluxo do l -ésimo nível no arco $(i, j) \in A$;

y_{ij}^l - variável binária que assume os valores 1 ou 0, dependendo do arco (i, j) estar ou não suportando fluxo do l -ésimo nível;

f_i - custo fixo (não-negativo) de escolha do nó $i \in R^l$ para prover fluxo do l -ésimo nível;

z_i - variável binária que assume os valores 1 ou 0, dependendo do nó $i \in R^l$ estar ou não provendo fluxo do l -ésimo nível;

M^l - capacidade dos arcos para suportar fluxo do l -ésimo nível, mas relaxada nesse trabalho e considerada um número “grande”, *i.e.* $M^l = \sum_{L=l}^m \sum_{i \in D^L} d_i$;

s^l - capacidade em todos os candidatos a nós de oferta do l -ésimo nível, também relaxada nesse trabalho, *i.e.* $s^l = M^l$;

$\delta^+(i)$ - conjunto $\{j \mid (i, j) \in A\}$;

$\delta^-(i)$ - conjunto $\{j \mid (j, i) \in A\}$.

O problema PRMN é formulado abaixo, como um modelo de programação matemática inteira-mista baseado em fluxos, (Cruz *et al.*, 1996c):

(N):

$$\min \sum_{l=1}^m \left[\sum_{(i,j) \in A} (c_{ij}^l x_{ij}^l + f_{ij}^l y_{ij}^l) + \sum_{i \in R^l} f_i z_i \right], \quad (3.1)$$

s.a.:

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = - \left(\sum_{j \in \delta^+(i)} x_{ij}^{l-1} - \sum_{j \in \delta^-(i)} x_{ji}^{l-1} \right), \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=2,3,\dots,m, \end{array} \quad (3.2)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall \quad \begin{array}{l} i \in T^l, \\ l=1,2,\dots,m, \end{array} \quad (3.3)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = -d_i, \quad \forall \quad \begin{array}{l} i \in D^l, \\ l=1,2,\dots,m, \end{array} \quad (3.4)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l \leq s^l z_i, \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=1,2,\dots,m, \end{array} \quad (3.5)$$

$$x_{ij}^l \leq M^l y_{ij}^l, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\dots,m, \end{array} \quad (3.6)$$

$$x_{ij}^l \geq 0, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\dots,m, \end{array} \quad (3.7)$$

$$y_{ij}^l \in \{0, 1\}, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\dots,m, \end{array} \quad (3.8)$$

$$z_i \in \{0, 1\}, \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=1,2,\dots,m. \end{array} \quad (3.9)$$

A função objetivo (3.1) minimiza três termos a saber: (i) o custo variável total para todos os m níveis, (ii) o custo fixo total associado à escolha dos arcos e (iii) o custo fixo total associado à escolha dos candidatos a nós de oferta.

As restrições (3.2) asseguram que haja conservação de fluxo em cada candidato a nó de oferta entre níveis adjacentes. As restrições (3.3) e (3.4) são as usuais igualdades de conservação de fluxos em cada nó de transbordo e em cada nó de demanda. As restrições (3.5) asseguram que não haja transformação de fluxos nos candidatos a nó de oferta, a menos que eles tenham sido selecionados. Finalmente, as restrições (3.6) expressam o fato de que o fluxo em um arco seja nulo, a menos que esse arco tenha sido incluído na topologia.

Entretanto, propomos uma formulação alternativa, mais conveniente aos nossos propósitos de descrever os algoritmos desenvolvidos. A formulação é capaz de representar o problema após a aplicação do teste de redução desenvolvido, bem como os subproblemas criados durante a busca na árvore de pesquisa, pelo algoritmo *branch-and-bound*, de forma semelhante ao que foi apresentado no Capítulo 2. Definamos $J_0 \subseteq N$, como o conjunto de nós que foram rejeitados na solução ótima por algum algoritmo qualquer, $J_1 \subseteq N$, o

conjunto de nós aceites e $J = N \setminus (J_0 \cup J_1)$, o conjunto de nós em estado indefinido. De modo similar, definamos $K_0^l \subseteq A$, como o conjunto de arcos rejeitados no l -ésimo nível, $K_1^l \subseteq A$, como o conjunto de arcos aceites no l -ésimo nível e $K^l = A \setminus (K_0^l \cup K_1^l)$, como o conjunto de arcos livres no l -ésimo nível. Assim, podemos representar o problema PRMN pela seguinte formulação, (Cruz *et al.*, 1996c):

(N'):

$$\min \sum_{l=1}^m \left[\sum_{(i,j) \in A} (c_{ij}^l x_{ij}^l + f_{ij}^l y_{ij}^l) + \sum_{i \in R^l} f_i z_i \right], \quad (3.10)$$

s.a.:

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = - \left(\sum_{j \in \delta^+(i)} x_{ij}^{l-1} - \sum_{j \in \delta^-(i)} x_{ji}^{l-1} \right), \quad \forall \quad \begin{array}{l} i \in R^l, \\ l=2,3,\dots,m, \end{array} \quad (3.11)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall \quad \begin{array}{l} i \in T^l, \\ l=1,2,\dots,m, \end{array} \quad (3.12)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = -d_i, \quad \forall \quad \begin{array}{l} i \in D^l, \\ l=1,2,\dots,m, \end{array} \quad (3.13)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l \leq s^l z_i, \quad \forall \quad \begin{array}{l} i \in R^l \cap J, \\ l=1,2,\dots,m, \end{array} \quad (3.14)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l \leq s^l, \quad \forall \quad \begin{array}{l} i \in R^l \cap J_1, \\ l=1,2,\dots,m, \end{array} \quad (3.15)$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall \quad \begin{array}{l} i \in R^l \cap J_0, \\ l=1,2,\dots,m, \end{array} \quad (3.16)$$

$$x_{ij}^l \leq M^l y_{ij}^l, \quad \forall \quad \begin{array}{l} (i,j) \in K^l, \\ l=1,2,\dots,m, \end{array} \quad (3.17)$$

$$x_{ij}^l \leq M^l, \quad \forall \quad \begin{array}{l} (i,j) \in K_1^l, \\ l=1,2,\dots,m, \end{array} \quad (3.18)$$

$$x_{ij}^l = 0, \quad \forall \quad \begin{array}{l} (i,j) \in K_0^l, \\ l=1,2,\dots,m, \end{array} \quad (3.19)$$

$$x_{ij}^l \geq 0, \quad \forall \quad \begin{array}{l} (i,j) \in A, \\ l=1,2,\dots,m, \end{array} \quad (3.20)$$

$$y_{ij}^l \in \{0,1\}, \quad \forall \quad \begin{array}{l} (i,j) \in K^l, \\ l=1,2,\dots,m, \end{array} \quad (3.21)$$

$$y_{ij}^l = 1, \quad \forall \quad \begin{array}{l} (i,j) \in K_1^l, \\ l=1,2,\dots,m, \end{array} \quad (3.22)$$

$$y_{ij}^l = 0, \quad \forall \quad \begin{array}{l} (i,j) \in K_0^l, \\ l=1,2,\dots,m, \end{array} \quad (3.23)$$

$$z_i \in \{0,1\}, \quad \forall \quad \begin{array}{l} i \in R^l \cap J, \\ l=1,2,\dots,m, \end{array} \quad (3.24)$$

$$z_i = 1, \quad \forall \quad \begin{array}{l} i \in R^l \cap J_1, \\ l=1,2,\dots,m, \end{array} \quad (3.25)$$

$$z_i = 0, \quad \forall \quad \begin{array}{l} i \in R^l \cap J_0, \\ l=1,2,\dots,m. \end{array} \quad (3.26)$$

3.3 Algoritmo *Branch-and-Bound*

O problema PRMN é \mathcal{NP} -árduo, uma vez que generaliza outros problemas de otimização \mathcal{NP} -árduos, como o problema de *Steiner* em grafos, (Garey e Johnson, 1979), ou ainda o problema de localização não capacitada, (Erlenkotter, 1978). Há duas formas de lidar com problemas \mathcal{NP} -árduos.

Em uma delas, após reconhecer a aparente inevitabilidade da complexidade de tempo exponencial, os pesquisadores procuram obter a máxima melhoria possível sobre a busca exaustiva. Entre as técnicas mais amplamente empregadas, estão os algoritmos do tipo *branch-and-bound*, também conhecidos na literatura de ciência da computação como estratégias *dividir para conquistar*. Soluções parciais são geradas, usando poderosos procedimentos para cálculo de limites superior e inferior, para o reconhecimento daquelas que com certeza não poderiam conduzir a soluções completas. Assim, em um simples passo, ramos inteiros da árvore de pesquisa são eliminados. Além disso, em alguns casos é possível conseguir uma redução na complexidade de tempo no pior caso, simplesmente fazendo uma escolha mais adequada dos objetos sobre os quais a busca exaustiva é executada.

Para aqueles casos onde o método exato é computacionalmente inviável, a alternativa é tentar encontrar heurísticas que produzam uma “boa” solução, dentro de um tempo de processamento aceitável. Um exemplo de técnica largamente empregada é a de busca local, na qual são usadas operações pré-estabelecidas, para a obtenção de melhorias em uma solução inicial qualquer, *e.g. simulated annealing*, (Aarts e Korst, 1989), e busca TABU, (Glover, 1989, Glover, 1990). Embora raramente seja possível dizer, através de uma análise *a priori* de uma heurística, qual seria o seu desempenho na prática, alguns resultados têm sido obtidos nessa área, (Altinkemer e Gavish, 1988, Klein e Ravi, 1993, Barrera *et al.*, 1993, Goemans *et al.*, 1994, Agrawal *et al.*, 1995). Tais resultados são conhecidos como prova formal de garantia de desempenho, que é uma vasta área de pesquisa.

Nesse Capítulo, devemos nos concentrar apenas na primeira alternativa. Um esboço de uma versão recursiva de um algoritmo *branch-and-bound* com busca em profundidade (*depth-first search*) é apresentado na Figura 3.1. O algoritmo enumera implicitamente todas as possibilidades de solução, escolhendo a melhor delas. É ineficiente, do ponto de vista computacional, devido à sua complexidade exponencial de tempo, $O(2^{m|J| + \sum_{l=1}^m |K^l|})$, mas pode ser útil na prática, resolvendo problemas de tamanho razoável, em tempos aceitáveis.


```

algorithm Solve( $N'$ )
  /* bounding */
  Compute_Lower_and_Upper_Bounds( $L, U$ )
  Update_Best_Upper_Bound( $U, U_{\text{BEST}}$ )
  GAP  $\leftarrow \frac{U-L}{U}$ 
  /* branching */
  if  $L > U_{\text{BEST}}$  then
    write 'Infeasible node reached.'
  else if GAP  $\leq \varepsilon$  then
    write 'Optimum reached.'
  else
    Reduce( $N'$ )
    VAR  $\leftarrow$  Chosen_Var_*
    if VAR is node  $i$  then
       $J \leftarrow J \setminus \{i\}; J_1 \leftarrow J_1 \cup \{i\}$ 
      Solve( $N'$ )
       $J_1 \leftarrow J_1 \setminus \{i\}; J_0 \leftarrow J_0 \cup \{i\}$ 
      Solve( $N'$ )
       $J_0 \leftarrow J_0 \setminus \{i\}; J \leftarrow J \cup \{i\}$ 
    else if VAR is arc  $(i, j)$  in the  $l$ -th level then
       $K^l \leftarrow K^l \setminus \{(i, j)\}; K_1^l \leftarrow K_1^l \cup \{(i, j)\}$ 
      Solve( $N'$ )
       $K_1^l \leftarrow K_1^l \setminus \{(i, j)\}; K_0^l \leftarrow K_0^l \cup \{(i, j)\}$ 
      Solve( $N'$ )
       $K_0^l \leftarrow K_0^l \setminus \{(i, j)\}; K^l \leftarrow K^l \cup \{(i, j)\}$ 
    else
      write 'Leaf reached.'
    end if
  end if
  end if
  end if
  Unreduce( $N'$ )
end if
end if
end if
end algorithm

```

Figura 3.1: Algoritmo *Branch-and-Bound* para o Problema PRMN

3.3.1 Cálculo dos Limites L e U

Por razões já discutidas anteriormente, usamos o algoritmo de otimização por subgradientes apresentado no Capítulo 2, Figura 2.3, para melhoria dos limites inferiores e superiores, que são calculados conforme o esquema seguinte.

Limites Inferiores

Há muitas possibilidades para derivação de uma relaxação lagrangeana para o modelo (N') . Propomos relaxar as restrições (3.14) usando os multiplicadores de Lagrange $v_i \geq 0$ e as restrições (3.20), usando os multiplicadores $w_{ij}^l \geq 0$. A função lagrangeana abaixo segue:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{z}; \mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) = & \sum_{l=1}^m \left[\sum_{(i,j) \in A} (c_{ij}^l x_{ij}^l + f_{ij}^l y_{ij}^l) + \sum_{i \in R^l} f_i z_i \right] + \\ & \sum_{l=1}^m \sum_{i \in R^l \cap J} v_i \left(\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l - s^l z_i \right) + \\ & \sum_{l=1}^m \sum_{(i,j) \in K^l} w_{ij}^l (x_{ij}^l - M^l y_{ij}^l), \end{aligned} \quad (3.27)$$

o que resulta na seguinte relaxação lagrangeana:

$(LR_{\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m})$:

$$L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) = \min_{\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m \geq 0} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{z}; \mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m), \quad (3.28)$$

s.a.:

$$\begin{aligned} & (3.11), (3.12), (3.13), (3.15), (3.16), (3.17), (3.18), \\ & (3.19), (3.21), (3.22), (3.23), (3.24), (3.25), (3.26). \end{aligned}$$

Supondo que $L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) = \mathcal{L}(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*; \mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m)$, um vetor subgradiente subgradiente γ da função L no ponto $(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m)$ é:

$$\gamma = \left\{ \left[\left(\sum_{j \in \delta^+(i)} x_{ij}^{l*} - \sum_{j \in \delta^-(i)} x_{ji}^{l*} \right) - s^l z_i^* \right]_{\substack{i \in R^l \cap J, \\ l=1,2,\dots,m}}, \left[x_{ij}^{l*} - M^l y_{ij}^{l*} \right]_{\substack{(i,j) \in K^l, \\ l=1,2,\dots,m}} \right\}. \quad (3.29)$$

Uma vez que valores viáveis tenham sido fornecidos para os multiplicadores de Lagrange $\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m$, o cálculo de $L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m)$ é reduzido à resolução de subproblemas “fáceis”¹:

¹Nesse contexto, “fácil” é usado em referência a problemas que podem ser resolvidos por algoritmos polinomiais com o tamanho da entrada.

$$\begin{aligned}
L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) &= L_1(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) + \\
&L_2(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) + \\
&L_3(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m),
\end{aligned} \tag{3.30}$$

onde $L_1(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m)$, $L_2(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m)$ e $L_3(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m)$ são as soluções ótimas dos problemas mostrados abaixo.

- O problema (L_1) é:

(L_1) :

$$L_1(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) = \min \sum_{l=1}^m \sum_{(i,j) \in A} C_{ij}^l x_{ij}^l, \tag{3.31}$$

s.a.:

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = - \left(\sum_{j \in \delta^+(i)} x_{ij}^{l-1} + \sum_{j \in \delta^-(i)} x_{ji}^{l-1} \right), \quad \forall \begin{array}{l} i \in R^l, \\ l=2,3,\dots,m, \end{array} \tag{3.32}$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall \begin{array}{l} i \in T^l, \\ l=1,2,\dots,m, \end{array} \tag{3.33}$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = -d_i, \quad \forall \begin{array}{l} i \in D^l, \\ l=1,2,\dots,m, \end{array} \tag{3.34}$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l \leq s^l, \quad \forall \begin{array}{l} i \in R^l \cap J_1, \\ l=1,2,\dots,m, \end{array} \tag{3.35}$$

$$\sum_{j \in \delta^+(i)} x_{ij}^l - \sum_{j \in \delta^-(i)} x_{ji}^l = 0, \quad \forall \begin{array}{l} i \in R^l \cap J_0, \\ l=1,2,\dots,m, \end{array} \tag{3.36}$$

$$x_{ij}^l \leq M^l, \quad \forall \begin{array}{l} (i,j) \in K_1^l, \\ l=1,2,\dots,m, \end{array} \tag{3.37}$$

$$x_{ij}^l = 0, \quad \forall \begin{array}{l} (i,j) \in K_0^l, \\ l=1,2,\dots,m, \end{array} \tag{3.38}$$

$$x_{ij}^l \geq 0, \quad \forall \begin{array}{l} (i,j) \in A, \\ l=1,2,\dots,m, \end{array} \tag{3.39}$$

onde

$$C_{ij}^l = \begin{cases} +\infty & (i, j) \in K_0^l, \\ c_{ij}^l & (i, j) \in K_1^l, \quad j \notin R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + v_i & (i, j) \in K_1^l, \quad j \notin R^l \cap J, \quad i \in R^l \cap J, \\ c_{ij}^l - v_j, & (i, j) \in K_1^l, \quad j \in R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + v_i - v_j, & (i, j) \in K_1^l, \quad j \in R^l \cap J, \quad i \in R^l \cap J, \\ c_{ij}^l + w_{ij}^l & (i, j) \in K^l, \quad j \notin R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + w_{ij}^l + v_i & (i, j) \in K^l, \quad j \notin R^l \cap J, \quad i \in R^l \cap J, \\ c_{ij}^l + w_{ij}^l - v_j, & (i, j) \in K^l, \quad j \in R^l \cap J, \quad i \notin R^l \cap J, \\ c_{ij}^l + w_{ij}^l + v_i - v_j, & (i, j) \in K^l, \quad j \in R^l \cap J, \quad i \in R^l \cap J. \end{cases} \quad (3.40)$$

A solução ótima de (L_1) pode ser calculada pelo algoritmo de caminhos mínimos. O problema pode ser resolvido por níveis. O ótimo do primeiro nível é conectar os nós $i \in D^1$ aos nós $j \in R^1$, pelos caminhos mais curtos, não usando os arcos do conjunto K_0^1 . Para o segundo nível, o ótimo é conectar os nós $i \in D^2$ também aos nós $j \in R^1$, mas usando necessariamente algum nó $k \in R^2$ e não usando os arcos pertencentes aos conjuntos K_0^1 e K_0^2 , e assim sucessivamente, para os níveis remanescentes. O algoritmo completo é visto na Figura 3.2.

O algoritmo de caminhos mínimos com custos arbitrários possui uma complexidade de tempo de pior caso $O(|N||A|)$, considerando que não há circuitos com custo negativo, (Bazaraa *et al.*, 1990). Assim, do Teorema 3.1 apresentado abaixo, (Cruz *et al.*, 1996a, Cruz *et al.*, 1996c), aquele algoritmo da Figura 3.2, se eficientemente implementado, tem complexidade de tempo polinomial $O(m|N||A|)$ no pior caso, o que significa $O(m|N|^3)$, em redes densas.

Teorema 3.1 *O problema (L_1) , sobre o dígrafo $\mathcal{D} = (N, A)$, com pesos definidos conforme a Equação (3.40), não possui circuitos com custos negativos.*

Prova (por construção): Denotemos por C^l o circuito arbitrário do l -ésimo nível. Denotemos por $A(C^l) \subseteq A$ o conjunto de arcos desse circuito. Finalmente, denotemos por $N(C^l) \subseteq N$ o conjunto de nós do circuito. Da Equação (3.40), o custo por unidade de fluxo do l -ésimo nível associado ao circuito C^l deve ser não-negativo:

$$\begin{aligned} \sum_{(i,j) \in A(C^l)} C_{ij}^l &= \sum_{(i,j) \in A(C^l)} c_{ij}^l + \sum_{(i,j) \in A(C^l) \cap K^l} w_{ij}^l + \sum_{i \in N(C^l) \cap R^l \cap J} v_i - \sum_{j \in N(C^l) \cap R^l \cap J} v_j \\ &= \sum_{(i,j) \in A(C^l)} c_{ij}^l + \sum_{(i,j) \in A(C^l) \cap K^l} w_{ij}^l \\ &\geq 0. \end{aligned}$$

■

- O Problema (L_2) é o seguinte problema de seleção:

```

procedure Solve_L1
    /*  $\sigma_i^l$  is the minimum per unit cost to bring */
    /*  $l$ -th level flow from set  $R^l$  to node  $i \in N$ ; */
    /* function SH( $i, j, l$ ) returns the shortest */
    /* path length from  $i$  to  $j$  using costs  $C_{ij}^l$ ; */
    L1  $\leftarrow$  0
    for all  $j \in R^1$  do
        if  $j \in J$  then
             $\sigma_j^0 \leftarrow$  0
        else
             $\sigma_j^0 \leftarrow +\infty$ 
        end if
    end for
    for  $l \leftarrow 1$  to  $m$  do
        for all  $j \in D^l$  do
             $\sigma_j^l \leftarrow \min_{i \in R^l} [\sigma_i^{l-1} + \text{SH}(i, j, l)]$ 
            L1  $\leftarrow$  L1 +  $\sigma_j^l * d_j$ 
        end for
        if  $l \neq m$  do
            for all  $j \in R^{l+1}$  do
                if  $j \in J$  then
                     $\sigma_j^l \leftarrow \min_{i \in R^l} [\sigma_i^{l-1} + \text{SH}(i, j, l)]$ 
                else
                     $\sigma_j^l \leftarrow +\infty$ 
                end if
            end for
        end if
    end for
    return L1
end procedure

```

Figura 3.2: Algoritmo para Resolução do Problema (L_1)

(L_2):

$$L_2(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) = \min \sum_{l=1}^m \sum_{(i,j) \in A} (f_{ij}^l - w_{ij}^l M^l) y_{ij}^l, \quad (3.41)$$

s.a.:

$$y_{ij}^l \in \{0, 1\}, \quad \forall \quad \begin{matrix} (i, j) \in K^l, \\ l=1,2,\dots,m, \end{matrix} \quad (3.42)$$

$$y_{ij}^l = 1, \quad \forall \quad \begin{matrix} (i, j) \in K_1^l, \\ l=1,2,\dots,m, \end{matrix} \quad (3.43)$$

$$y_{ij}^l = 0, \quad \forall \quad \begin{matrix} (i, j) \in K_0^l, \\ l=1,2,\dots,m, \end{matrix} \quad (3.44)$$

que pode ser resolvido por um algoritmo com complexidade de tempo $O(m|A|)$, ou $O(m|N|^2)$, em redes densas.

• Similarmente, o Problema (L_3) é também um problema de seleção:

(L_3):

$$L_3(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) = \min \sum_{l=1}^m \sum_{i \in R^l} (f_i - v_i s^l) z_i, \quad (3.45)$$

s.a.:

$$z_i \in \{0, 1\}, \quad \forall \quad \begin{matrix} i \in R^l \cap J, \\ l=1,2,\dots,m, \end{matrix} \quad (3.46)$$

$$z_i = 1, \quad \forall \quad \begin{matrix} i \in R^l \cap J_1, \\ l=1,2,\dots,m, \end{matrix} \quad (3.47)$$

$$z_i = 0, \quad \forall \quad \begin{matrix} i \in R^l \cap J_0, \\ l=1,2,\dots,m, \end{matrix} \quad (3.48)$$

que pode ser resolvido por um algoritmo com complexidade de tempo $O(|N|)$.

Limites Superiores

O objetivo aqui é propor um procedimento para determinar soluções viáveis de maneira rápida e simplificada. Há muitas maneiras de calcular um limite superior para o modelo (N'). Propomos usar a solução ótima do problema (L_1), que pode ser convertida em uma solução primal para (N'), se for assegurada a viabilidade das restrições relaxadas (3.14) e (3.20). Para isso, basta calcular os custos dos fluxos usando os custos c_{ij}^l originais e acrescentar os custos fixos f_{ij}^l e f_i respectivos. A heurística proposta é claramente polinomial.

3.3.2 Escolha das Variáveis para *Branching*

A escolha da primeira variável de decisão livre é a estratégia mais simples, Figura 3.3. É fácil de ver que a complexidade de tempo do algoritmo, no pior caso, é $O(m(|N| + |A|))$.

O Teorema 3.2, abaixo, (Cruz *et al.*, 1996a), estende o Teorema 2.1, (Cruz *et al.*, 1996b), estabelecendo uma importante propriedade das soluções ótimas dos problemas PRMN:

```

procedure Chosen_Var_First_Free
  for  $l = 1$  to  $m$  do
    /* try to choose a node */
    for all  $i \in R^l$  do
      if  $i \in J$  then
        return  $i$ 
      end if
    end for
    /* otherwise try to choose an arc */
    for all  $(i, j) \in A$  do
      if  $(i, j) \in K^l$  then
        return  $(i, j), l$ 
      end if
    end for
  end for
  return FAIL
end procedure

```

Figura 3.3: Estratégia Simplificada de *Branching* para o Problema PRMN

Teorema 3.2 *Se o problema PRMN, conforme a formulação (N), possuir uma solução ótima, então existe um conjunto ótimo de arcos tal que sobre os nós incida, no máximo, um arco de cada nível.*

Prova (por contradição): *Suponhamos que o Teorema 3.2 não seja satisfeito por nenhuma solução ótima e que o nó u tenha dois arcos incidentes do l -ésimo nível, digamos (s, u) e (t, u) . Deverá existir um conjunto de arcos na solução ótima que forme um caminho sem ciclos, de algum dos candidatos a nós de oferta do l -ésimo nível ao nó u , passando pelo nó s , i.e. usando o arco (s, u) . Esse caminho será denominado P_s . De modo similar, deverá haver também um caminho sem ciclos usando o arco (t, u) , que será denominado P_t . Sem perda de generalidade, vamos supor que*

$$\sum_{(i,j) \in P_s} c_{ij}^l \leq \sum_{(i,j) \in P_t} c_{ij}^l.$$

Assim, desabilitando o arco (t, u) e transferindo o seu fluxo x_{tu} do caminho P_t ao caminho P_s , haverá, no mínimo, a seguinte redução no valor da função objetivo:

$$\left(\sum_{(i,j) \in P_t} c_{ij}^l - \sum_{(i,j) \in P_s} c_{ij}^l \right) x_{ij} + f_{tu} \geq 0.$$

A solução resultante dessa transferência é no mínimo tão boa quanto a solução original e satisfaz o Teorema 3.2, contradizendo a nossa suposição inicial. ■

```

procedure Chosen_Var_No-Cycling
  /* try to choose a first level candidate supply node */
  for all  $i \in R^1$  do
    if  $i \in J$  then
      return  $i$ 
    end if
  end for
  /* otherwise go throughout all levels */
  for  $l = 1$  to  $m$  do
    /* try to choose a  $(l + 1)$ -th level candidate supply node */
    for all  $i \in R^{l+1}$  do
      if  $i \in J$  then
        return  $i$ 
      end if
    end for
    /* otherwise try to choose an arc in the  $l$ -th level */
    /* compute set of reached nodes  $E$  */
     $E \leftarrow \emptyset$ 
    for all  $i \in N$  do
      if  $i \in R^l \cap J_1$  then
         $E \leftarrow E \cup \{i\}$ 
      end if
    end for
    for all  $(i, j) \in A$  do
      if  $(i, j) \in K_1^l$  then
         $E \leftarrow E \cup \{i\} \cup \{j\}$ 
      end if
    end for
    /* search new eligible free arc */
    if there is  $i \in D^l \cup (R^{l+1} \cap J_1)$  such that  $i \notin E$  then
      for all  $(i, j) \in A$  do
        if  $(i, j) \in K^l$  and  $i \in E$  and  $j \notin E$  then
          return  $(i, j), l$ 
        end if
      end for
    end if
  end for
  return FAIL
end procedure

```

Figura 3.4: Estratégia Melhorada de *Branching* para o Problema PRMN

Baseado no Teorema 3.2, podemos sugerir uma estratégia melhorada de escolha de variável para *branching*, (Cruz *et al.*, 1996a), Figura 3.4. A escolha do candidato a nó de oferta é decidida em, no máximo, $O(|N|)$ operações. A computação do conjunto T pode ser feita $O(|N| + |A|)$. A escolha de um arco é decidida em, no máximo, $O(|N| + |A|)$ operações, notando que o teste “**if there is** $i \in D^l \cup (R^{l+1} \cap J_1)$ **such that** $i \notin T$ ”, Figura 3.4, pode ser feito $O(|N|)$. Assim, isso resulta em uma complexidade de tempo no pior caso $O(m(|N| + |A|))$, igual, portanto, àquela do procedimento apresentado na Figura 3.3.

3.3.3 Algoritmo de Redução

Propomos o algoritmo apresentado na Figura 3.5, (Cruz *et al.*, 1996a), para redução do problema PRMN. Esse procedimento é uma extensão daquele apresentado no Capítulo 2, Figura 2.6. No pior caso, o laço “**repeat comandos until condição**”, Figura 3.5, deverá terminar após $O(|N| + m|A|)$ iterações, considerando que apenas uma redução ocorre a cada iteração. O comando “**for all** $i \in R^l \cap J$ **do comandos end for**” é $O(|N|m|N||A|)$, uma vez que envolve $O(|N|)$ cálculos de limites inferiores, que são $O(m|N||A|)$. De modo análogo, o comando “**for all** $(i, j) \in K^l$ **do comandos end for**” é $O(|A|m|N||A|)$. Assim, no pior caso, o algoritmo de redução tem complexidade de tempo polinomial, que é:

$$O \left[m^2 |N| |A| (|N| + m|A|) (|N| + |A|) \right].$$

Relembrando o algoritmo *branch-and-bound* apresentado na Figura 3.1, o procedimento de restabelecimento, Figura 3.6, (Cruz *et al.*, 1996a), deverá ser ativado no estágio de *backtracking*. O procedimento também tem complexidade de tempo polinomial, $O(|N| + m|A|)$, no pior caso.

3.3.4 Definição das Estruturas de Dados

As estruturas de dados precisam ser projetadas com cuidado, para assegurarmos o bom desempenho dos algoritmos. O tipo abstrato de dados principal é o dígrafo multi-ponderado $\mathcal{D} = (N, A)$. Várias formas de representação poderiam ser empregadas nesse caso e a escolha apropriada depende das operações necessárias e da quantidade de informação a ser armazenada, (Aho *et al.*, 1983, Ziviani, 1993).

A Figura 3.7 ilustra a representação do problema PRMN por lista de adjacência. A Figura A.1, no Apêndice A, apresenta um fragmento de código em C , para sua criação. Note que a alocação do espaço necessário é feita em tempo de execução, eliminando a necessidade de definições de tamanhos máximos e novas recompilações a cada vez que aparece um problema maior do que o esperado.

Outra definição importante é de como representar os conjuntos R^l and D^l . Listas encadeadas simples são uma escolha aceitável, uma vez que as operações do tipo “**for all** $i \in R^l$, $l = 1, 2, \dots, m$. **do comandos end for**” são bem suportadas. A Figura 3.8 ilustra a representação que utilizamos. A Figura A.2, no Apêndice A, apresenta um fragmento de código em C usado para sua implementação. Como no caso anterior, é usada alocação dinâmica.

```

procedure Reduce( $M'$ )
    /* initialize set of nodes and arcs recently fixed */
     $F_J \leftarrow \emptyset$ ;  $F_{K^l} \leftarrow \emptyset$ ,  $l = 1, 2, \dots, m$ 
    /* proceed with reduction */
    repeat
        for  $l = 1$  to  $m$  do
            /* try to fix free supply nodes */
            for all  $i \in R^l \cap J$  do
                 $J \leftarrow J \setminus \{i\}$ ;  $J_1 \leftarrow J_1 \cup \{i\}$ 
                if  $L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) > U_{\text{BEST}}$  then
                     $J_1 \leftarrow J_1 \setminus \{i\}$ ;  $J_0 \leftarrow J_0 \cup \{i\}$ ;  $F_J \leftarrow F_J \cup \{i\}$ 
                else
                     $J_1 \leftarrow J_1 \setminus \{i\}$ ;  $J_0 \leftarrow J_0 \cup \{i\}$ 
                    if  $L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) > U_{\text{BEST}}$  then
                         $J_0 \leftarrow J_0 \setminus \{i\}$ ;  $J_1 \leftarrow J_1 \cup \{i\}$ ;  $F_J \leftarrow F_J \cup \{i\}$ 
                    else
                         $J_0 \leftarrow J_0 \setminus \{i\}$ ;  $J \leftarrow J \cup \{i\}$ 
                    end if
                end if
            end for
            /* try to fix free arcs */
            for all  $(i, j) \in K^l$  do
                 $K^l \leftarrow K^l \setminus \{(i, j)\}$ ;  $K_1^l \leftarrow K_1^l \cup \{(i, j)\}$ 
                if  $L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) > U_{\text{BEST}}$  then
                     $K_1^l \leftarrow K_1^l \setminus \{(i, j)\}$ ;  $K_0^l \leftarrow K_0^l \cup \{(i, j)\}$ ;  $F_{K^l} \leftarrow F_{K^l} \cup \{(i, j)\}$ 
                else
                     $K_1^l \leftarrow K_1^l \setminus \{(i, j)\}$ ;  $K_0^l \leftarrow K_0^l \cup \{(i, j)\}$ 
                    if  $L(\mathbf{v}, \mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^m) > U_{\text{BEST}}$  then
                         $K_0^l \leftarrow K_0^l \setminus \{(i, j)\}$ ;  $K_1^l \leftarrow K_1^l \cup \{(i, j)\}$ ;  $F_{K^l} \leftarrow F_{K^l} \cup \{(i, j)\}$ 
                    else
                         $K_0^l \leftarrow K_0^l \setminus \{(i, j)\}$ ;  $K^l \leftarrow K^l \cup \{(i, j)\}$ 
                    end if
                end if
            end for
        end for
    until no reduction has occurred
end procedure

```

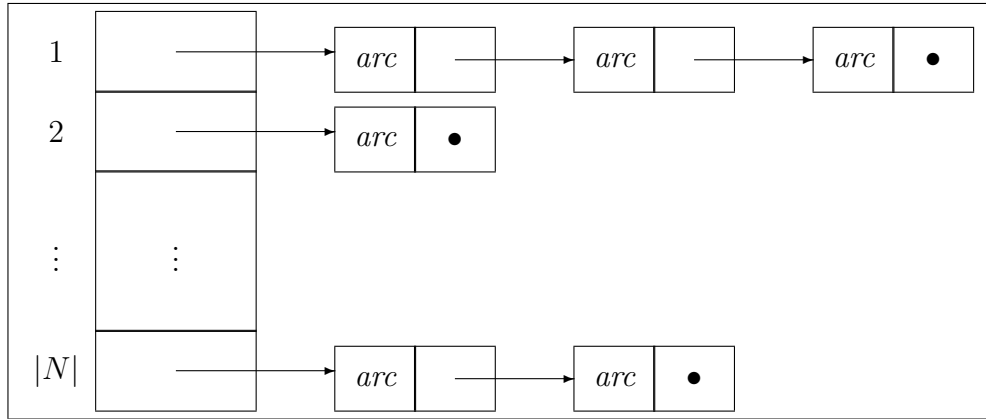
Figura 3.5: Algoritmo de Redução para o Problema PRMN

```

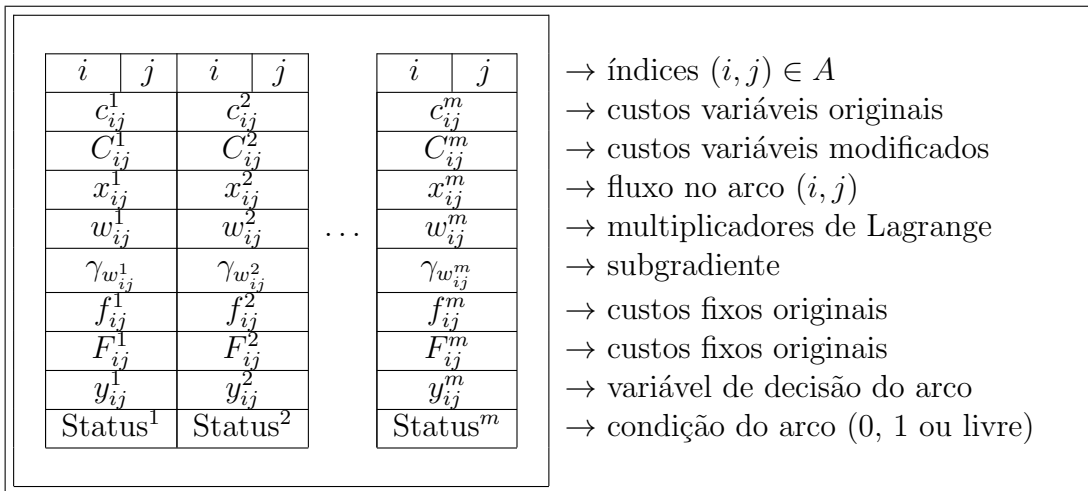
procedure Unreduce( $M'$ )
  /* reset nodes */
  for all  $i \in F_J$  do
    if  $i \in J_0$  then
       $J_0 \leftarrow J_0 \setminus \{i\}$ 
    else
       $J_1 \leftarrow J_1 \setminus \{i\}$ 
    end if
     $J \leftarrow J \cup \{i\}$ 
  end for
  /* reset arcs */
  for  $l = 1$  to  $m$  do
    for all  $(i, j) \in F_{K^l}$  do
      if  $(i, j) \in K_0^l$  then
         $K_0^l \leftarrow K_0^l \setminus \{(i, j)\}$ 
      else
         $K_1^l \leftarrow K_1^l \setminus \{(i, j)\}$ 
      end if
       $K^l \leftarrow K^l \cup \{(i, j)\}$ 
    end for
  end for
end procedure

```

Figura 3.6: Procedimento de Restabelecimento para o Problema PRMN

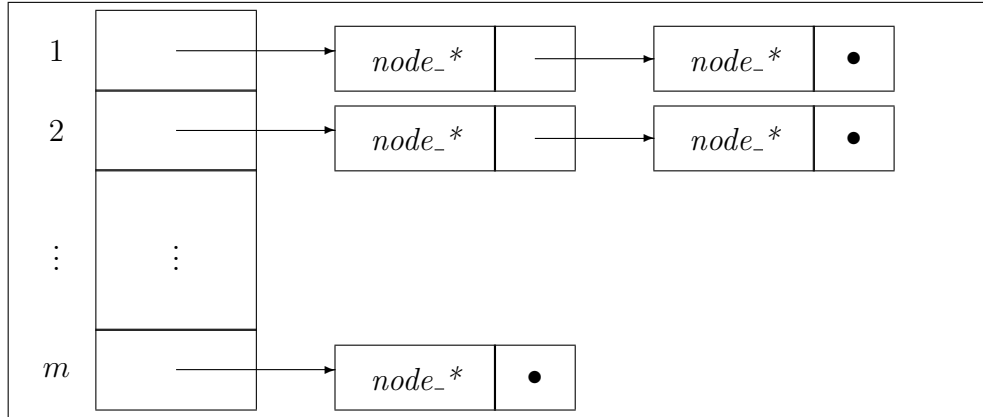


a) Lista de Adjacências

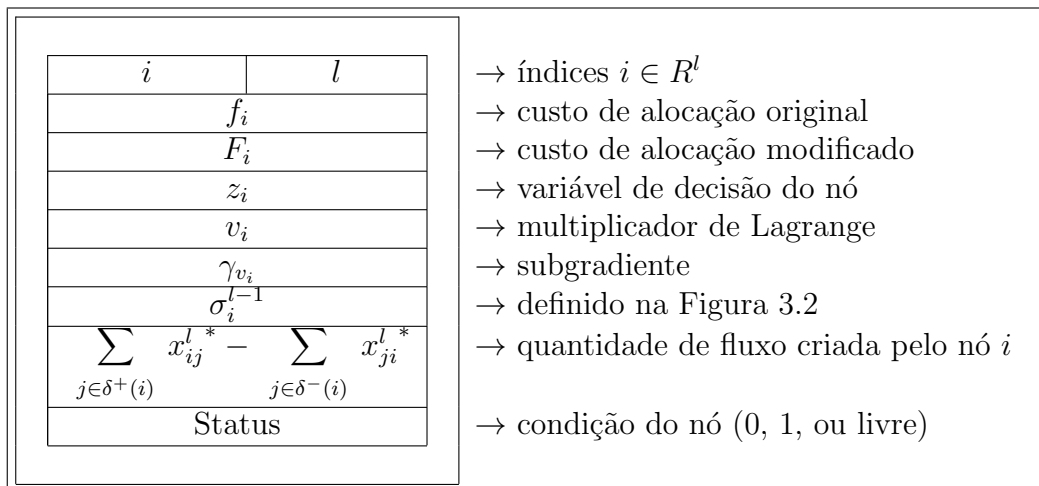


b) Definição do Elemento *arc*

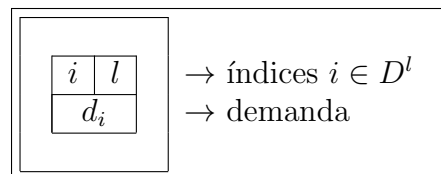
Figura 3.7: Representação do Dígrafo $\mathcal{D} = (N, A)$ por Listas de Adjacências Multi-Ponderadas



a) Lista Encadeada Simples para um Nó Genérico $node_*$



b) Definição do Elemento $node_root$



c) Definição do Elemento $node_demand$

Figura 3.8: Representação dos Conjuntos R^l e D^l , $l = 1, 2, \dots, m$, por Listas Encadeadas Simples

Como observação final, o conjunto de nós alcançados E , Figura 3.4, precisa ser implementado como um vetor com $|N|$ posições, pois tal implementação suporta as operações do tipo “ $i \in T?$ ” e “ $T \leftarrow T \cup \{i\}$ ” com complexidade constante $O(1)$.

3.4 Experiência Computacional

Uma versão preliminar de todos os algoritmos foi codificada em C e está disponível a pedido. Todos os testes foram executados em uma DECstation 3100 com o sistema operacional ULTRIX V4.2A (Rev. 47). Usamos o parâmetro ε das Figuras 3.1 e 2.3 igual a 10^{-6} , por ser essa a precisão do compilador, para números reais representados pelo tipo *float*.

Os problemas foram gerados pelo seguinte algoritmo. Usando a distribuição uniforme, $|N|$ nós foram alocados em um quadrado (100×100) e $(|N| - 1)$ arestas foram definidas, garantindo uma rede conexa. Arestas adicionais foram então definidas, até que um número total de $|A|/2$ fosse alcançado. Finalmente, o grafo indireto resultante foi convertido em um dígrafo, pela substituição de cada vértice por dois arcos de sentidos opostos. Os pesos Ω_{ij} nos arcos foram definidos como a distância euclideana entre as extremidades i e j . Esse procedimento segue uma orientação conhecida, (Aneja, 1980), e tem sido utilizado extensivamente para geração de casos de teste para algoritmos para o problema de *Steiner* em grafos, (Wong, 1984, Beasley, 1989, Duin e Volgenant, 1989b).

A distribuição uniforme foi empregada também para a composição dos conjuntos D^l e R^l , $l = 1, 2, \dots, m$. Os parâmetros $|N|$, $|A|$, $|R^1|$, $|D^1|$, $|R^2|$, $|D^2|$, e f_i assumiram valores diversos, na tentativa de cobrir uma ampla gama de redes diferentes. O objetivo é produzir uma tabela de referência, com a qual seria possível estimar o tempo de execução para algum outro problema qualquer que alguém quisesse resolver. Os custos f_{ij}^l e c_{ij}^l efetivamente usados foram derivados dos pesos Ω_{ij} , usando as constantes apresentadas nas tabelas de resultados. Todas as demandas foram consideradas unitárias. Os custos no nível superior foram considerados menores do que os custos no nível inferior. Essa é uma suposição razoável em redes em multi-níveis. Usualmente, os níveis mais altos tiram proveito das demandas mais concentradas e podem empregar meios especiais, com custos mais baixos. Por outro lado, os níveis inferiores, frequentemente, têm que usar meios menos eficientes e portanto mais caros.

A Tabela 3.1 apresenta uma comparação entre as duas estratégias de *branching*, para aqueles casos onde tivemos tempos inferiores a 30.000 segundos. Os resultados apresentados são o número de nós gerados pelo algoritmo *branch-and-bound*, até a otimalidade, e o tempo gasto em segundos. Todas as operações de entrada e saída foram desconsideradas e apenas um processo estava na máquina àquele instante. A estratégia melhorada de *branching*, Figura 3.4, teve um desempenho bem superior, mas mesmo a estratégia simplificada, Figura 3.3, não possui um desempenho medíocre, se comparada à enumeração explícita, conforme apresentado na Tabela 3.2. Podemos notar que o algoritmo *branch-and-bound* associado à relaxação lagrangeana permite um excelente ganho em termos do número de possibilidades explicitamente avaliadas.

A Tabela 3.3 apresenta o efeito notável do algoritmo de redução, associado à estratégia

melhorada de *branching* da Figura 3.4. Instâncias de tamanhos maiores tornaram-se agora tratáveis.

Como observação final, parece-nos que a relaxação lagrangeana, embora oferecendo *gaps* elevados, é uma excelente heurística para o problema PRMN. Como pode ser visto na Tabela 3.4, a solução obtida no primeiro nó da árvore de pesquisa foi a ótima em todos os casos testados. A exploração teve que prosseguir apenas para provar a otimalidade da solução.

Tabela 3.1: Efeito da Estratégia de *Branching* no Problema PRMN

N	A	R ¹	D ¹	R ²	D ²	$\frac{f_{ij}^1}{\Omega_{ij}}$	$\frac{c_{ij}^1}{\Omega_{ij}}$	$\frac{f_{ij}^2}{\Omega_{ij}}$	$\frac{c_{ij}^2}{\Omega_{ij}}$	f_i	Chosen_Var_																		
											First_Free		No_Cycling																
											Nós	UCP(s)	Nós	UCP(s)															
4	12	1	0	1	2	1	4	2	8	0	587	30,00	25	1,60															
										128	587	30,00	25	1,50															
										1.024	587	30,00	25	1,50															
										2	1	1	4	2	8	0	1	0,01	1	0,01									
																128	1	0,01	1	0,01									
																1.024	1	0,01	1	0,01									
										8	14	1	0	1	2	1	4	2	8	0	1.123	80,00	61	4,90					
																				128	1.123	80,00	61	4,90					
																				1.024	1.123	80,00	61	5,00					
																				4	1	4	2	8	0	7.279	510,00	81	6,40
																									128	7.279	520,00	81	6,40
																									1.024	7.279	510,00	81	6,40
2	2	1	4	2	8	0	567	42,00	89											7,70									
						128	351	27,00	45											3,90									
						1.024	351	27,00	45											3,90									
4	1	4	2	8	0	31.555	2.400,00	225	19,00																				
					128	20.313	1.600,00	147	12,00																				
					1.024	19.311	1.500,00	69	5,90																				
4	2	1	4	2	8	0	7.105	600,00	657											61,00									
						128	1.757	160,00	65											6,00									
						1.024	1.757	160,00	65											6,00									
28	1	0	1	2	1	4	2	8	0											987	170,00	17	3,50						
									128											987	160,00	17	3,50						
									1.024											987	170,00	17	3,50						
									4											1	4	2	8	0	**	**	41	8,40	
																								128	**	**	41	8,40	
																								1.024	**	**	41	8,40	
									2											2	1	4	2	8	0	**	**	71	15,00
																									128	**	**	71	15,00
																									1.024	**	**	81	18,00
									4	1	4	2	8	0	**	**	369	77,00											
														128	**	**	413	86,00											
														1.024	**	**	499	100,00											
									4	2	1	4	2	8	0	**	**	1,801	400,00										
															128	**	**	161	38,00										
															1.024	**	**	91	22,00										

**Não disponível (*overflow* de tempo)

Tabela 3.2: Ganho do *Branch-and-Bound* sobre a Enumeração Explícita no Problema PRMN

$ N $	$ A $	$ R^1 $	$ D^1 $	$ R^2 $	$ D^2 $	Enumeração Explícita	<i>Chosen_Var_First_Free</i>
						Nós = $2^{2 A + R^1 + R^2 }$	Nós
4	12	1	0	1	2	$6,7 \times 10^7$	587
				2	1	$1,3 \times 10^8$	1
8	14	1	0	1	2	$1,7 \times 10^9$	1.123
					4	$1,7 \times 10^9$	7.279
				2	2	$2,1 \times 10^9$	567
					4	$2,1 \times 10^9$	31.555
				4	2	$8,6 \times 10^9$	7.105
28	1	0	1	2	2	$2,9 \times 10^{17}$	987
					4	$2,9 \times 10^{17}$	**
				2	2	$5,8 \times 10^{17}$	**
					4	$5,8 \times 10^{17}$	**
				4	2	$2,3 \times 10^{18}$	**

**Não disponível (*overflow* de tempo)

3.5 Conclusões

Nesse Capítulo, apresentamos o problema PRMN e reforçamos a importância do modelo para aplicações práticas. O problema é formulado e um algoritmo do tipo *branch-and-bound* baseado na relaxação lagrangeana é detalhado. São derivados uma nova estratégia de *branching*, baseada em uma propriedade das soluções ótimas, e um novo teste de redução, baseado na relaxação lagrangeana. Embora considerado computacionalmente ineficiente devido ao seu tempo de processamento no pior caso ser exponencial com o tamanho da entrada, o algoritmo *branch-and-bound* pode ser muito útil na prática, para pequenas instâncias do problema PRMN, conforme nossos resultados atestam. Além disso, é provado que a estratégia de *branching* e o teste de redução tornam possível a resolução de instâncias maiores.

Continuam em aberto questões tais como se existiriam melhores métodos para geração dos limites usados pelo algoritmo *branch-and-bound*, melhores estratégias de *branching* ou mesmo testes de redução mais poderosos, como aqueles conhecidos para o problema de *Steiner* em grafos, (Maculan *et al.*, 1991, Duin e Volgenant, 1989b), ou para alguns problemas de planejamento de redes hierárquicas, (Duin e Volgenant, 1989a). Trabalhos futuros poderiam explorar essas questões e também o desenvolvimento de heurísticas polinomiais, talvez com alguma garantia teórica de desempenho, na linha do que já é conhecido para um caso particular do problema PRMN, (Balakrishnan *et al.*, 1994c). Finalmente, vale a pena mencionar que com a ênfase crescente em robustez e confiabilidade, (Balakrishnan *et al.*, 1994b, Gendreau *et al.*, 1995), estudos envolvendo modelos melhorados, com restrições de conectividade, são uma área promissora para pesquisas futuras.

Tabela 3.3: Efeito do Algoritmo de Redução no Problema PRMN

N	A	R ¹	D ¹	R ²	D ²	$\frac{f_{ij}^1}{\Omega_{ij}}$	$\frac{c_{ij}^1}{\Omega_{ij}}$	$\frac{f_{ij}^2}{\Omega_{ij}}$	$\frac{c_{ij}^2}{\Omega_{ij}}$	f_i	Sem Redução		Com Redução								
											Nós	UCP(s)	Nós	UCP(s)							
8	14	1	0	1	2	1	4	2	8	0	61	4,90	1	0,22							
										128	61	4,90	1	0,23							
										1.024	61	4,90	1	0,22							
							4	1	4	2	8	0	81	6,40	1	0,22					
						128						81	6,40	1	0,22						
						1.024						81	6,40	1	0,22						
						2	2	1	4	2	8	0	89	7,70	3	0,29					
					128							45	3,90	1	0,24						
					1.024							45	3,90	1	0,24						
						4	1	4	2	8	0	225	19,00	3	0,25						
					128						147	12,00	1	0,25							
					1.024						69	5,90	1	0,25							
	4	2	1	4	2	8	0	657	61,00	3	0,28										
128							65	6,00	1	0,27											
1.024							65	6,00	1	0,28											
							28	1	0	1	2	1	4	2	8	0	17	3,50	1	0,53	
128																17	3,50	1	0,54		
1.024																17	3,50	1	0,53		
	4	1	4	2	8	0	41	8,40	1	0,63											
128						41	8,40	1	0,63												
1.024						41	8,40	1	0,63												
						2	2	1	4	2	8	0	71	15,00	1	0,68					
128												71	15,00	1	0,67						
1.024												81	18,00	3	1,80						
	4	1	4	2	8	0	369	77,00	3	2,20											
128						413	88,00	5	3,30												
1.024						499	100,00	3	2,30												
						4	2	1	4	2	8	0	1.801	400,00	29	17,00					
128												161	38,00	13	7,80						
1.024												91	22,00	3	2,00						
	16	30	1	0	4							2	1	4	2	8	0	3.405	880,00	3	0,88
128																	167	45,00	1	0,88	
1.024																	33	10,00	1	0,86	
						4	1	4	2	8	0						7.035	1.800,00	3	0,88	
128											1.645						420,00	1	1,00		
1.024											87						25,00	3	2,40		
	8	1	4	2	8	0	25.797	6.800,00	3	0,90											
128						16.381	4.300,00	1	1,00												
1.024						109	32,00	3	1,90												
						8	2	1	4	2	8	0	**	**	3	1,20					
128												**	**	1	1,30						
1.024												**	**	1	1,20						
	4	1	4	2	8	0	**	**	3	1,10											
128						**	**	1	1,20												
1.024						**	**	1	1,10												
	60	1	0	4	2	1	4	2	8	0	**	**	3	2,50							
128										**	**	1	2,90								
1.024										**	**	3	5,50								
										4	1	4	2	8	0	**	**	29	52,00		
128															**	**	7	17,00			
1.024															**	**	3	7,50			
	8	1	4	2	8	0	**	**	16.479	30.000,00											
128						**	**	215	450,00												
1.024						**	**	55	120,00												

**Não disponível (*overflow* de tempo)

Tabela 3.4: Soluções no Primeiro Nó para o Problema PRMN

$ N $	$ A $	$ R^1 $	$ D^1 $	$ R^2 $	$ D^2 $	$\frac{f_{ij}^1}{\Omega_{ij}}$	$\frac{c_{ij}^1}{\Omega_{ij}}$	$\frac{f_{ij}^2}{\Omega_{ij}}$	$\frac{c_{ij}^2}{\Omega_{ij}}$	f_i	SOL*	GAP	k^\dagger	UCP(s)					
8	14	1	0	1	2	1	4	2	8	0	1,000	2,30	216	0,19					
										128	1,000	2,10	216	0,19					
										1.024	1,000	1,40	216	0,19					
									4	1	4	2	8	0	1,000	2,70	206	0,19	
					128	1,000	2,60	206						0,19					
					1.024	1,000	2,00	207						0,19					
									2	2	1	4	2	8	0	1,000	2,60	228	0,22
					128	1,000	2,30	228							0,21				
					1.024	1,000	1,20	228							0,21				
									4	1	4	2	8	0	1,000	6,50	218	0,24	
					128	1,000	6,10	217						0,21					
					1.024	1,000	4,20	218						0,24					
									4	2	1	4	2	8	0	1,000	4,90	236	0,23
					128	1,000	4,30	236							0,24				
					1.024	1,000	2,50	236							0,24				
28	1	0	1	2	1	4	2	8	0	1,000	5,30	323	0,52						
									128	1,000	4,20	323	0,46						
									1.024	1,000	1,70	323	0,46						
									4	1	4	2	8	0	1,000	12,00	318	0,47	
					128	1,000	11,00	314						0,46					
					1.024	1,000	6,30	313						0,46					
									2	2	1	4	2	8	0	1,000	11,00	325	0,48
					128	1,000	20,00	330							0,49				
					1.024	1,000	12,00	335							0,50				
									4	1	4	2	8	0	1,000	15,00	320	0,48	
					128	1,000	19,00	324						0,49					
					1.024	1,000	16,00	325						0,49					
									4	2	1	4	2	8	0	1,000	11,00	338	0,54
					128	1,000	16,00	343							0,73				
					1.024	1,000	7,00	343							0,55				
16	30	1	0	4	2	1	4	2	8	0	1,000	6,40	359	0,70					
										128	1,000	5,80	359	0,70					
										1.024	1,000	3,60	359	0,70					
									4	1	4	2	8	0	1,000	9,00	344	0,67	
					128	1,000	11,00	344						0,67					
					1.024	1,000	13,00	349						0,69					
									8	1	4	2	8	0	1,000	7,20	339	0,68	
					128	1,000	8,30	347						0,70					
					1.024	1,000	13,00	351						0,72					
									8	2	1	4	2	8	0	1,000	7,70	374	0,82
					128	1,000	10,00	374							0,82				
					1.024	1,000	9,80	379							0,84				
									4	1	4	2	8	0	1,000	4,90	360	0,79	
					128	1,000	6,60	362						0,80					
					1.024	1,000	7,00	367						0,82					
60	1	0	4	2	1	4	2	8	0	1,000	7,50	588	1,90						
									128	1,000	13,00	593	2,00						
									1.024	1,000	9,40	598	2,00						
								4	1	4	2	8	0	1,000	13,00	580	1,90		
				128	1,000	16,00	578						2,00						
				1.024	1,000	14,00	585						1,90						
								8	1	4	2	8	0	1,000	13,00	569	1,90		
				128	1,000	15,00	568						1,90						
				1.024	1,000	23,00	581						1,90						

*SOL = $\frac{U_{BEST}}{U_{OPT}}$

†Iterações do algoritmo de otimização por subgradientes

Capítulo 4

Experiência Computacional com Implementações Paralelas do Algoritmo *Branch-and-Bound*

Nesse Capítulo, apresentamos resultados computacionais de implementações paralelas do clássico algoritmo *branch-and-bound* aplicado ao problema PRMN. As implementações são adequadas às arquiteturas paralelas de classificação MIMD (*Multiple Instruction stream, Multiple Data stream*). São, portanto, muito convenientes para o uso em redes de *workstations* (NOWs) que têm se tornado bem populares hoje em dia. Testamos duas versões: (i) centralizada e (ii) distribuída. Na versão distribuída, três estratégias de balanço de carga foram examinadas. Os resultados são encorajadores, apontando para um ganho sobre o processamento sequencial, em termos de tempo total de execução, (Cruz e Mateus, 1997).

4.1 Introdução

A idéia de redução do tempo de execução de algoritmos *branch-and-bound* via paralelização é bastante promissora, (Laursen, 1993). Há tanto abordagens usando a exploração paralela da árvore de pesquisa, quanto abordagens empregando o paralelismo para o cálculo dos próprios limites inferior e superior associados a cada um dos seus nós.

Nesse Capítulo, estamos preocupados em estudar o desempenho da primeira abordagem, que corresponde a uma aplicação paralela de granularidade mais grossa, onde o tamanho do grão é a quantidade relativa de trabalho executado entre duas sincronizações (comunicações) sucessivas. Para isso, usamos intensivamente o sistema SABOR, (Tavares, 1995, Tavares *et al.*, 1995), que fornece um ambiente de programação no qual o usuário fica liberado dos detalhes relativos à sincronização entre os processos, devendo concentrar-se apenas no desenvolvimento da sua aplicação.

Na Seção 4.2, descrevemos as implementações paralelas usadas para o algoritmo *branch-and-bound*. Na Seção 4.3, apresentamos e discutimos os resultados computacionais obtidos. As conclusões finais apresentadas na Seção 4.4 encerram esse Capítulo.

```

algorithm Branch-and-Bound
   $U_{\text{BEST}} \leftarrow +\infty$ 
   $\mathcal{L} \leftarrow \{(N')^0\}$ 
  while  $\mathcal{L} \neq \emptyset$  do
    /* search rule */
    select and delete a problem  $(N')^i$  from  $\mathcal{L}$ 
    /* bound rule */
    Compute_Lower_and_Upper_Bounds( $L^i, U^i$ )
    update  $U_{\text{BEST}}$ 
    /* branch rule */
    if  $L^i < U_{\text{BEST}}$  and  $(N')^i$  is not a leaf then
       $\mathcal{L} \leftarrow \mathcal{L} \cup \{(N')^{2i+1}\} \cup \{(N')^{2i+2}\}$ 
    end if
  end while
end algorithm

```

Figura 4.1: Algoritmo *Branch-and-Bound* Não-Recursivo

4.2 Algoritmos Implementados

As implementações paralelas usadas são baseadas na versão sequencial não-recursiva que apresentamos na Figura 4.1. Na descrição, U_{BEST} é o melhor limite superior e \mathcal{L} é a lista de problemas $(N')^i$ ainda não explorados, ver Capítulo 3, Seção 3.2. Cada problema inexplorado é da forma $Z_{N'}^i = \min\{\mathbf{c}\mathbf{x} \text{ s.a.: } \mathbf{x} \in S^i\}$, onde $S^i \subseteq S$ e S é o conjunto de soluções viáveis. Associado a cada problema em \mathcal{L} , há um limite inferior $L^i \leq Z_{N'}^i$ e um limite superior $U^i \geq Z_{N'}^i$.

A política de caminamento empregada é *last-in-first-out*, por medida de economia de memória, correspondendo a uma busca em profundidade (*depth-first search*). Os limites L^i e U^i são calculados segundo o procedimento mais eficiente descrito no Capítulo 3. Antes da geração dos filhos $(N')^{2i+1}$ e $(N')^{2i+2}$, os problemas $(N')^i$ são reduzidos. Para a escolha da variável de *branching*, adotamos a política que evita aqueles arcos que formam ciclos. Maiores detalhes sobre o algoritmo de redução e sobre o critério de escolha são apresentados no Capítulo 3.

O algoritmo *branch-and-bound* é paralelizado adotando-se um modelo mestre-escravo. O mestre é o responsável pelas inicializações gerais, pela criação dos escravos e sua coordenação. Em seguida, descrevemos as duas versões utilizadas nos nossos experimentos, a centralizada e a distribuída.

<pre> $U_{BEST} \leftarrow +\infty$ $\mathcal{L} \leftarrow \{(N')^0\}$ while <i>there is work</i> do if $\mathcal{L} \neq \emptyset$ then <i>select a problem $(N')^i$ from \mathcal{L}</i> <i>send $(N')^i$ and U_{BEST} to slave</i> end if if <i>there is an answer</i> then <i>receive U'_{BEST} and \mathcal{L}' from slave</i> <i>update U_{BEST}</i> $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}'$ end if end while <i>terminate slaves</i> </pre>	<pre> $U'_{BEST} \leftarrow +\infty$ while <i>not terminate</i> do <i>receive problem $(N')^i$ and U_{BEST}</i> <i>Compute_Lower_and_Upper_Bounds(L^i, U^i)</i> <i>update U'_{BEST}</i> if $L^i < U'_{BEST}$ and <i>$(N')^i$ is not a leaf</i> then $\mathcal{L}' \leftarrow \{(N')^{2i+1}\} \cup \{(N')^{2i+2}\}$ else $\mathcal{L}' \leftarrow \emptyset$ end if <i>send U'_{BEST} and \mathcal{L}' to master</i> end while </pre>
---	--

a) Algoritmo Mestre

b) Algoritmo Escravo

Figura 4.2: Implementação Centralizada do Algoritmo *Branch-and-Bound*

4.2.1 Versão Centralizada

Os algoritmos em alto nível da versão centralizada são apresentados na Figura 4.2. Nessa versão, o mestre administra a lista de problemas \mathcal{L} , mas a tarefa de expansão, *i.e.* o cálculo dos limites e a criação dos nós filhos, é atribuída aos escravos.

Em outras palavras, o mestre permanece no laço principal enquanto há problemas $(N')^i$ a serem expandidos na lista \mathcal{L} ou há algum escravo ainda em trabalho de expansão. Se houver algum problema na lista \mathcal{L} , o mestre escolhe um deles de acordo com a política de caminamento *last-in-first-out* e o transfere a um dos escravos sem trabalho. Se houver alguma expansão concluída, o mestre recebe o melhor limite superior parcial U'_{BEST} e a lista de filhos gerados \mathcal{L}' . Atualiza o seu melhor limite superior U_{BEST} e a sua lista \mathcal{L} . Caso contrário, o mestre manda uma ordem de término a todos os escravos. Os escravos, por sua vez, ficam em um laço principal recebendo problemas $(N')^i$, resolvendo-os e mandando os resultados ao mestre, até que chegue um sinal de término.

Pela necessidade frequente de sincronizações, essa implementação pode resultar em uma baixa taxa de uso dos processos escravos. Contudo, pode ser eficaz se a granularidade do problema for grossa o bastante, *i.e.* se a operação de expansão for computacionalmente intensa em comparação com o tempo de comunicação entre os processos.

A versão distribuída que apresentamos em seguida procura contornar essa possível desvantagem.

```

 $U_{\text{BEST}} \leftarrow +\infty$ 
generate problems  $(N')^{i_k}$ 
for all  $k$  do
    send problem  $(N')^{i_k}$  to slave  $k$ 
end for
while somebody is working do
    receive current status of slaves
end while
terminate slaves
for all  $k$  do
    receive  $U'_{\text{BEST}}$  from slave  $k$ 
    update  $U_{\text{BEST}}$ 
end for

```

```

receive problem  $(N')^{i_k}$ 
 $U'_{\text{BEST}} \leftarrow +\infty$ 
 $\mathcal{L} \leftarrow \{(N')^{i_k}\}$ 
while not terminate do
    if  $\mathcal{L} \neq \emptyset$  then
        select a problem  $(N')^i$  from  $\mathcal{L}$ 
        Compute_Lower_and_Upper_Bounds( $L^i, U^i$ )
        update  $U'_{\text{BEST}}$ 
        if  $L^i < U'_{\text{BEST}}$  and  $(N')^i$  is not a leaf then
             $\mathcal{L}' \leftarrow \{(N')^{2i+1}\} \cup \{(N')^{2i+2}\}$ 
        else
             $\mathcal{L}' \leftarrow \emptyset$ 
        end if
        send  $U'_{\text{BEST}}$  and  $\mathcal{L}'$  to other slaves
    end if
    receive  $U'_{\text{BEST}}$  and  $\mathcal{L}'$  from other slaves
     $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{L}'$ 
    send current status to master
end while
send results to master

```

a) Algoritmo Mestre

b) Algoritmo Escravo

Figura 4.3: Implementação Distribuída do Algoritmo *Branch-and-Bound*

4.2.2 Versão Distribuída

Os algoritmos em alto nível da versão distribuída são apresentados na Figura 4.3. O mestre tem a responsabilidade da criação e distribuição inicial de carga e da finalização do algoritmo. Cada escravo implementa o algoritmo *branch-and-bound* sequencial com pequenas modificações de modo a permitir trocas de cargas com outros escravos.

O mestre expande o problem $(N')^0$ até que tenha derivado tantos outros problemas $(N')^{i_k}$ a serem expandidos quanto são os escravos. Então, os distribui. Em seguida, permanece em um laço, aguardando que todos os escravos tenham terminado de explorar completamente os problemas $(N')^{i_k}$ recebidos. Então, termina todos os escravos, recebe os melhores limites superiores U'_{BEST} de cada um e atualiza o seu próprio melhor limite superior U_{BEST} .

De acordo com a política de balanceamento vigente, os escravos podem resolver os problemas $(N')^{i_k}$ recebidos fazendo ou não intercâmbio de cargas entre si. Testamos três estratégias de balanceamento diferentes, que descrevemos a seguir.

Balanceamento Estático

Esta é o procedimento mais simples e significa uma completa falta de balanceamento dinâmico. Cada escravo recebe um problema inicial $(N')^{i_k}$ e deve resolvê-lo completamente, sem nenhuma interação com os demais escravos. Possivelmente, essa estratégia apresentará um desempenho ruim em instâncias cuja árvore de pesquisa seja desbalanceada, *i.e.* instâncias cujos problemas $(N')^{i_k}$ apresentem diferentes graus de dificuldade. Voltamos ao assunto na Seção 4.3, onde apresentamos e discutimos os resultados computacionais.

Balanceamento de Karp-Zhang

Essa política de balanceamento é descrita no trabalho de Karp e Zhang, (Karp e Zhang, 1993). É também muito simples, mas os escravos podem trocar cargas entre si. Quando algum escravo expande um problema, ou pode manter os problemas gerados na sua própria lista de problemas \mathcal{L} , ou então pode mandá-los aos vizinhos, que são escolhidos por uma distribuição uniforme.

Nesse caso, o mestre comunica-se exclusivamente com um dos escravos e lhe manda o problema original $(N')^0$. Os demais escravos receberão cargas na medida em que algum escravo que já esteja trabalhando os selecione para mandar-lhes seus problemas gerados.

Balanceamento de Karp-Zhang Modificado

Na prática, acreditamos poder melhorar o algoritmo de Karp-Zhang acrescentando as seguintes modificações. Primeiro, um escravo não mandará a lista de filhos \mathcal{L}' a outro escravo, se isso resultar em uma carga local nula. Segundo, os escravos serão mais *ativos* e requisitarão cargas aos vizinhos, quando estiverem com a lista de problemas \mathcal{L} vazia. Esse não é um método tão simples quanto os anteriores, mas pode resultar em algum ganho, em termos de tempo de processamento.

A seguir, apresentamos os resultados computacionais, onde comparamos todas as possibilidades discutidas.

4.3 Apresentação e Discussão dos Resultados

Antes de mostrar os resultados computacionais, apresentamos as métricas empregadas para avaliar e comparar o desempenho das implementações. Várias são as medidas relevantes de desempenho de algoritmos paralelos, (Quinn, 1987). Lançamos mão de apenas duas delas, que descrevemos agora. A primeira é o *speedup* $s(p)$. Há várias formas de defini-lo, mas nesse trabalho usaremos apenas a seguinte:

$$s(p) = \frac{t_{\text{seq}}}{t_{\text{par}}(p)}, \quad (4.1)$$

onde t_{seq} é o tempo gasto pelo melhor algoritmo sequencial conhecido e $t_{\text{par}}(p)$ é o tempo gasto pelo algoritmo paralelo com p processadores.

Tabela 4.1: Detalhes de *Hardware* das Máquinas Utilizadas

Nome	Sistema Operacional	Release	Tipo da UCP	Modelo	Memória
diamante	SunOS	5.5	Model 61 SuperSPARC	Axil 320	256 MB
aroeira	SunOS	5.5.1	Model 140 UltraSPARC	Ultra 1	128 MB
turmalina	SunOS	5.5.1	Model 140 UltraSPARC	Ultra 1	160 MB
candeia	SunOS	5.5.1	110 MHz microSPARC II	SPARCstation 4	32 MB
caviuna	SunOS	5.5.1	110 MHz microSPARC II	SPARCstation 4	32 MB
cello	SunOS	5.5	50 MHz microSPARC I	SPARCclassic	16 MB
fluorita	SunOS	5.5	50 MHz microSPARC I	SPARCclassic	16 MB
azurita	SunOS	5.5	50 MHz microSPARC I	SPARCclassic	16 MB
aruak	SunOS	5.5	50 MHz microSPARC I	SPARCclassic	48 MB

Outra importante medida é a taxa de utilização u de cada processador, definida como se segue:

$$u = \frac{t_{\text{calc}}}{t_{\text{total}}} \times 100\%, \quad (4.2)$$

onde t_{total} é o tempo total de execução e t_{calc} é o tempo efetivamente gasto com processamento útil, *i.e.* o tempo total descontado o tempo em que o mestre ou o escravo estão em estado de espera (*idle state*).

Todos os resultados foram executados usando-se as máquinas disponíveis na rede de *workstations* do DCC-ICEX-UFMG, uma rede *Ethernet*, 10 Mbits, conectada através de barramento com protocolo TCP/IP, (Tanenbaum, 1981), e servidor de arquivos NFS. Maiores informações sobre o *hardware* de cada máquina, individualmente, são apresentadas na Tabela 4.1. Essas máquinas foram usadas como um computador paralelo *heterogêneo*, com rede de interconexão *completa*.

Por questão de simplicidade, testamos apenas problemas NCFCF, embora o sistema esteja pronto para resolver também problemas PRMN. As duas instâncias testadas B-3 e B-11 são derivados de uma biblioteca de problemas de domínio público, (Beasley, 1990), usada para testar algoritmos para os problema de *Steiner* em grafos, (Beasley, 1989, Duin e Volgenant, 1989b). Assim, algumas modificações devem ser feitas para adaptá-los ao problema NCFCF. Os pesos Ω_{ij} dos arcos dos problemas originais foram multiplicados pelos fatores constantes 1 e 10, de modo a gerar os custos fixos f_{ij} e variáveis c_{ij} , respectivamente. Um nó não-*Steiner* foi escolhido como centro de oferta e aos demais nós não-*Steiner* atribuímos demanda unitária.

Todos os resultados de *speedup* são baseados nos *menores* tempos sequenciais apresentados na Tabela 4.2. As máquinas estavam rodando exclusivamente o algoritmo sequencial e todas as operações de entrada e saída foram desconsideradas nos tempos apresentados.

Na Figura 4.4, mostramos os gráficos do *speedup* médio $s(p)$ obtidos para todas as versões testadas. Como não havia a possibilidade de usar as máquinas envolvidas nos testes exclusivamente para rodar o nosso algoritmo, as medidas de tempo foram tomadas sobre 5 (cinco) experimentos, em horários de baixa carga na rede, normalmente à noite e durante os finais de semana. Podemos observar que o *speedup* médio cresce com o

Tabela 4.2: Tempos Sequenciais Básicos

Máquina	Tempo de UCP em Segundos	
	Problema B-3 [†]	Problema B-11 [†]
	($ N = 50$, $ A = 126$ e $ D = 24$)	($ N = 75$, $ A = 300$ e $ D = 18$)
diamante	130.00	240.00
aroeira	78.00	110.00
turmalina	180.00	300.00
caviuna	140.00	210.00
candeia	140.00	210.00
cello	400.00	590.00
fluorita	400.00	590.00
azurita	400.00	590.00
aruak	790.00	1,200.00

[†]Redes de Beasley, (Beasley, 1990), com $f_{ij}/\Omega_{ij} = 1$ e $c_{ij}/\Omega_{ij} = 10$.

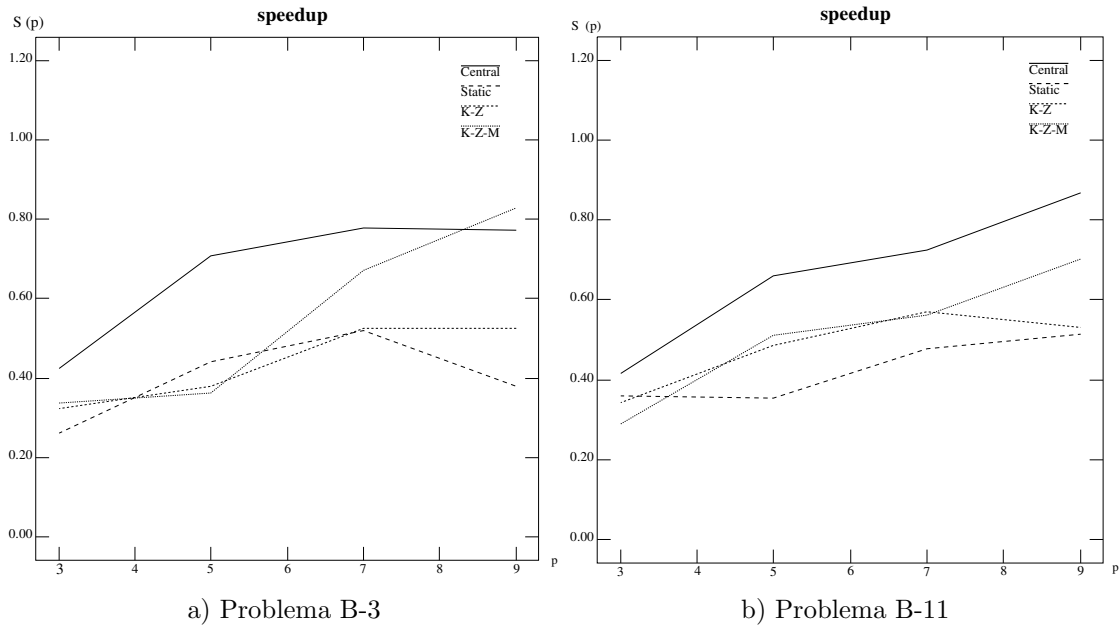
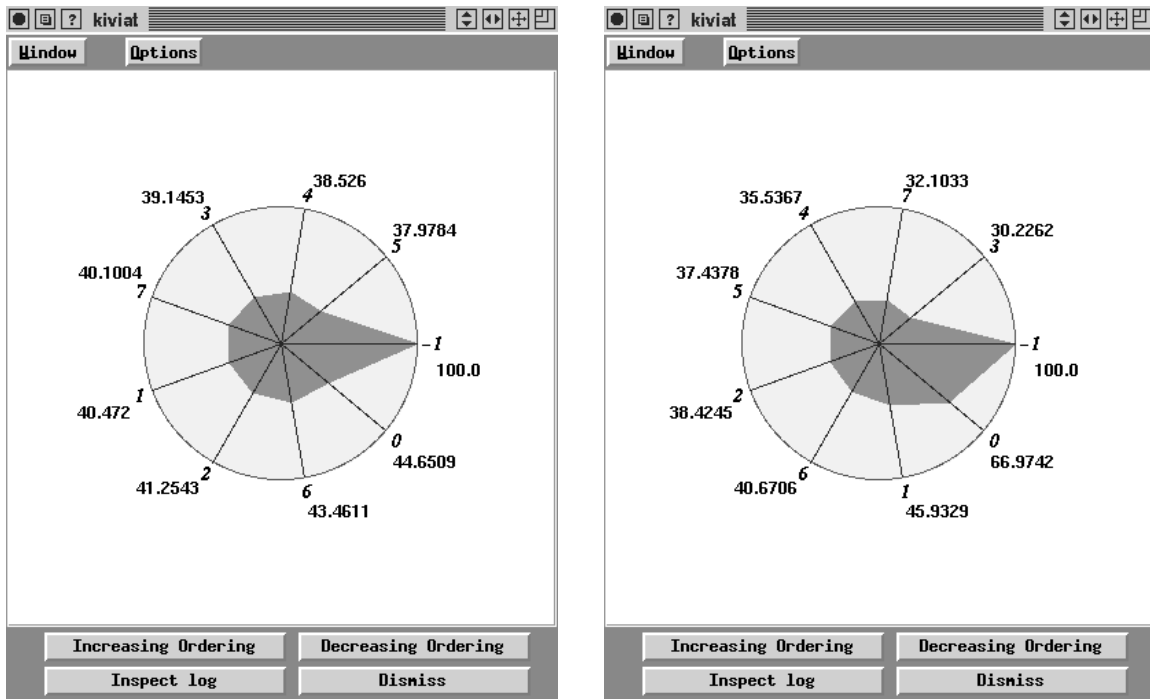


Figura 4.4: *Speedup* Médio para as Implementações Paralelas



a) Problema B-3

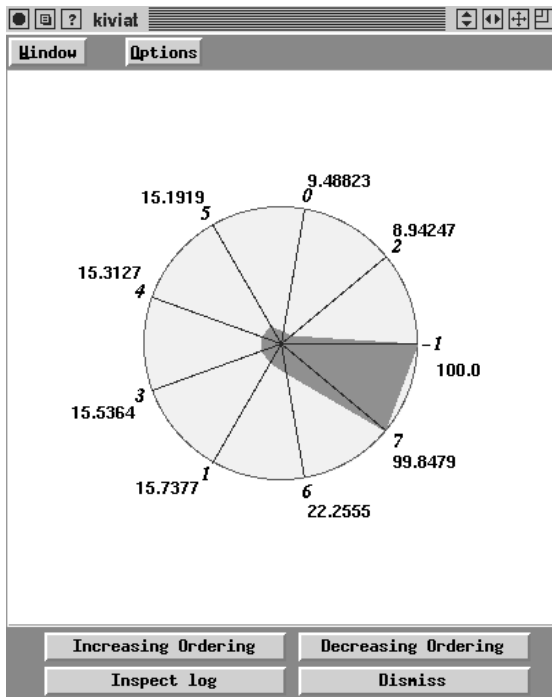
b) Problema B-11

Figura 4.5: Diagramas de Kiviat para a Versão Centralizada

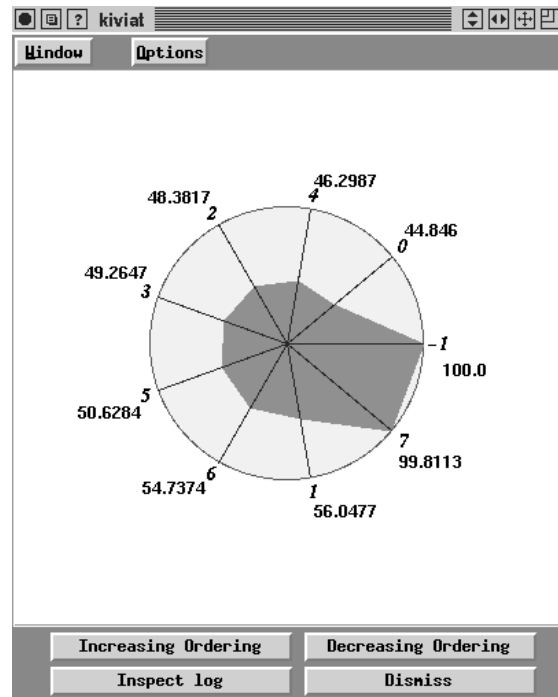
número de processadores, sendo aproximadamente linear, $O(p)$. Assim, é de se supor que se tivéssemos mais máquinas disponíveis, poderíamos conseguir tempos de processamento ainda menores.

Na Figura 4.5, mostramos os diagramas de Kiviat para a versão centralizada, no caso extremo de 9 (nove) processadores. Esses diagramas representam graficamente a taxa de utilização de cada um dos processadores. O processador identificado por “-1” está rodando o algoritmo mestre. Os demais processadores rodam o algoritmo escravo. Por esses diagramas, podemos observar que o uso dos processadores é baixo e a distribuição de carga pode ser um pouco irregular, dependendo da instância. Esse algoritmo parece ser adequado a problemas de granularidade mais grossa, *i.e.* problemas cujo tempo de cálculo dos limites e geração dos filhos seja maior do que o tempo de transmissão dos nós, de um processador a outro. De fato, o algoritmo apresenta um *speedup* melhor para o problema com mais variáveis inteiras, o problema B-11. Na implementação empregada, o algoritmo entrega a carga ao primeiro escravo que estiver sem trabalho. Para melhorar a distribuição de carga, talvez fosse interessante usar uma política de atribuição diferente.

Embora as sincronizações sejam menos frequentes na versão distribuída estática, o *speedup* médio não melhora, conforme pode ser visto pela Figura 4.4. O problema de distribuição não-uniforme de carga é ainda mais proeminente, conforme mostrado pela Figura 4.6. Além disso, o desempenho dessa versão é fortemente dependente da instância de entrada, o que é indesejável. A Figura 4.7 ilustra as árvores de pesquisa para os dois problemas e torna mais claro essa interdependência. De fato, podemos notar que



a) Problema B-3



b) Problema B-11

Figura 4.6: Diagramas de Kiviat para a Versão Distribuída Estática

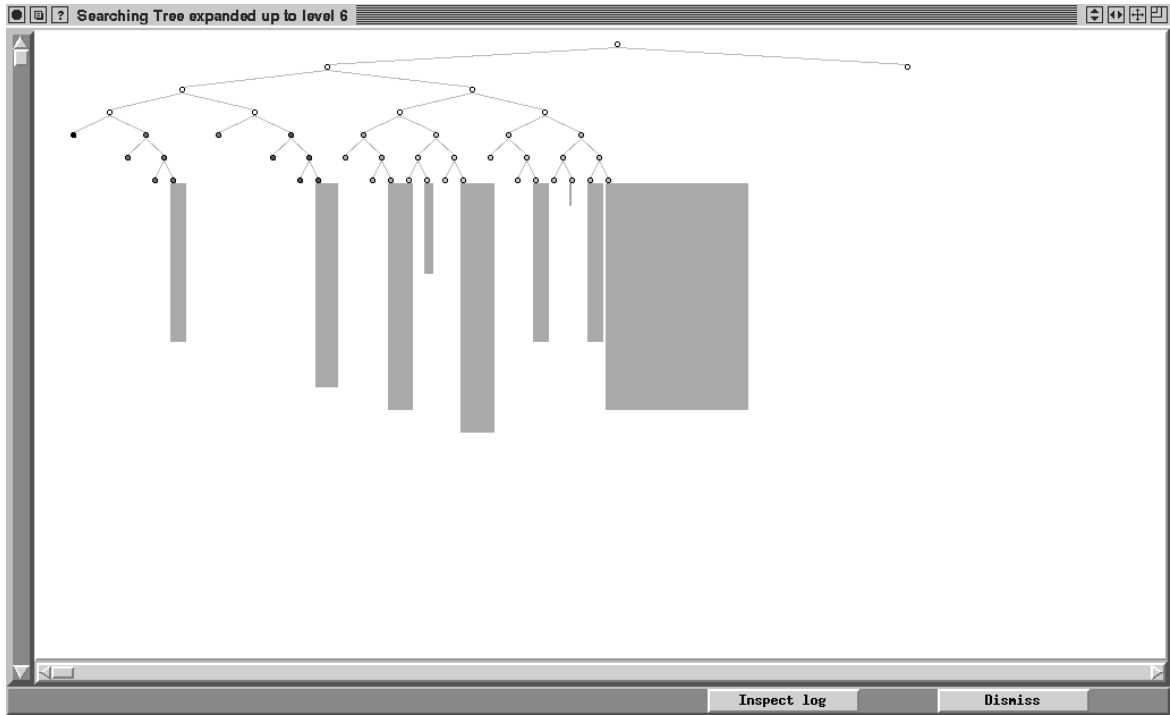
o problema com taxa de utilização mais equilibrada corresponde àquele com árvore de pesquisa mais balanceada, *i.e.* o problema B-11.

Casos de desbalanceamento como o mostrado anteriormente parecem ter sido a inspiração para métodos que utilizam trocas de cargas, como o método de balanceamento de Karp-Zhang. A Figura 4.4 apresenta a curva de *speedup* médio obtida para a versão distribuída com o balanceamento de Karp-Zhang. Embora o *speedup* não seja substancialmente diferente do obtido pela versão distribuída estática, podemos notar pela Figura 4.8 que houve uma melhoria na taxa de utilização. Ao contrário da versão estática, a distribuição para esse balanceamento parece ser razoavelmente independente da instância de entrada.

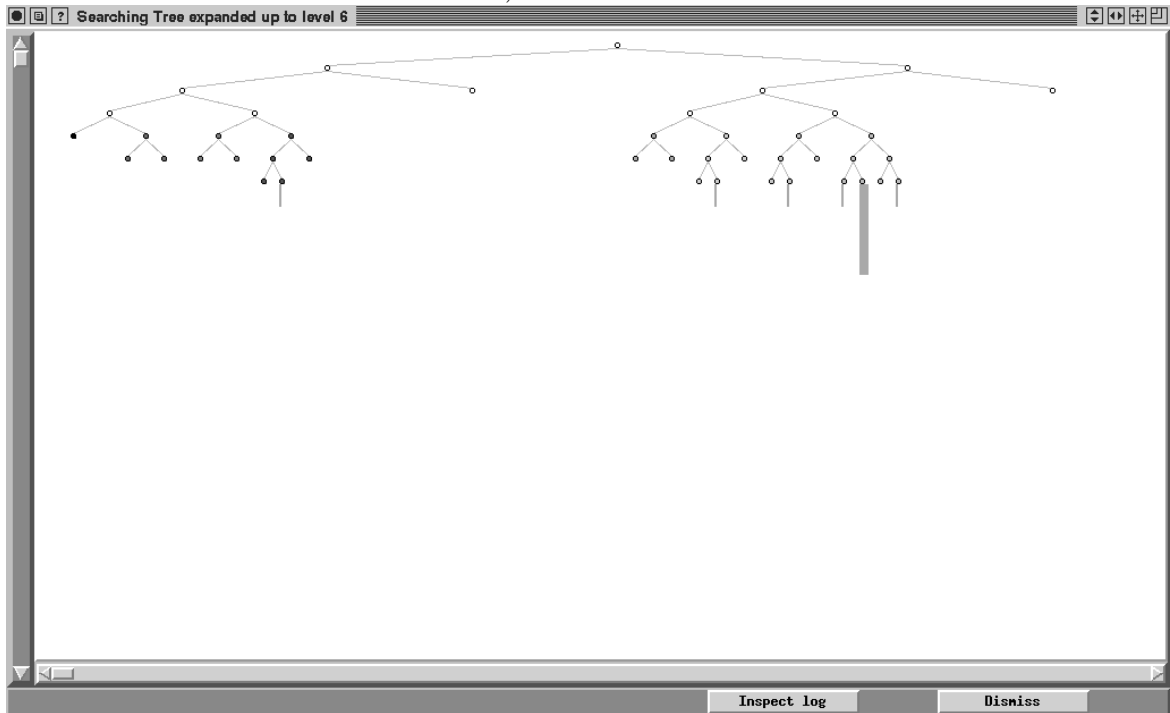
Mostramos nas Figuras 4.4 e 4.9 os resultados para a versão com o balanceamento de Karp-Zhang modificado. As modificações propostas parecem ser eficazes, uma vez que conseguimos melhores *speedups* e uma melhor taxa de utilização dos processadores. Além disso, obtivemos uma melhor distribuição de carga, também razoavelmente independente da instância de entrada.

4.4 Observações Finais

Apresentamos nesse Capítulo duas versões paralelas, centralizada e distribuída, para o clássico algoritmo *branch-and-bound*. Na versão distribuída, utilizamos três técnicas de

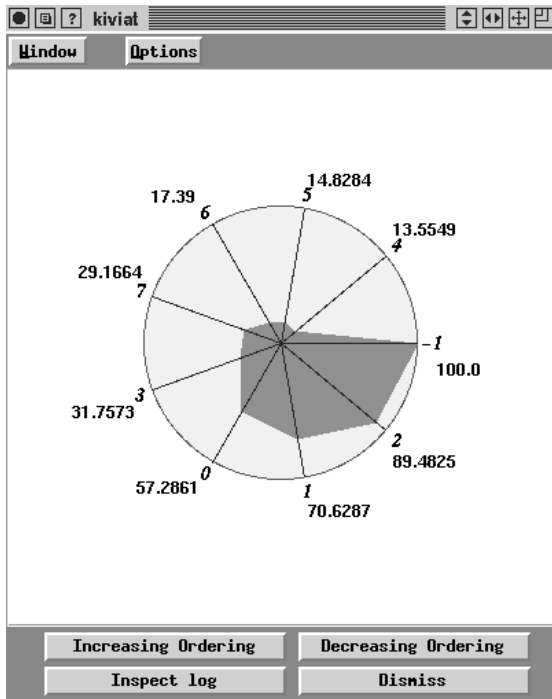


a) Problema B-3

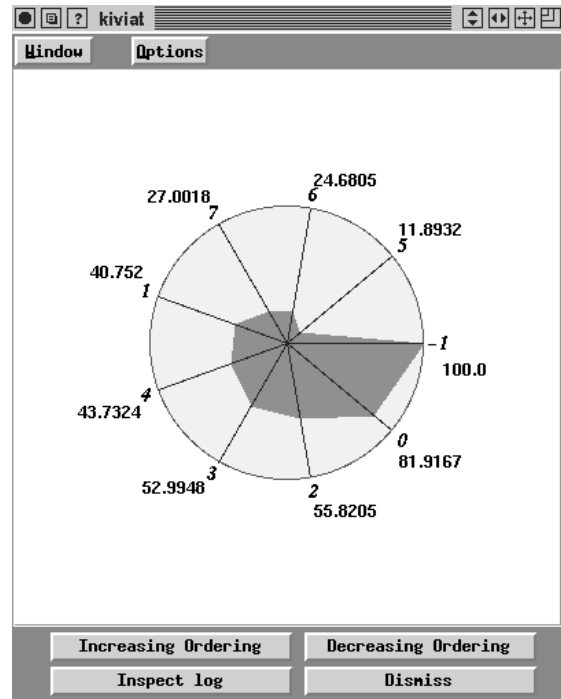


b) Problema B-11

Figura 4.7: Árvores de Pesquisa para a Versão Distribuída Estática

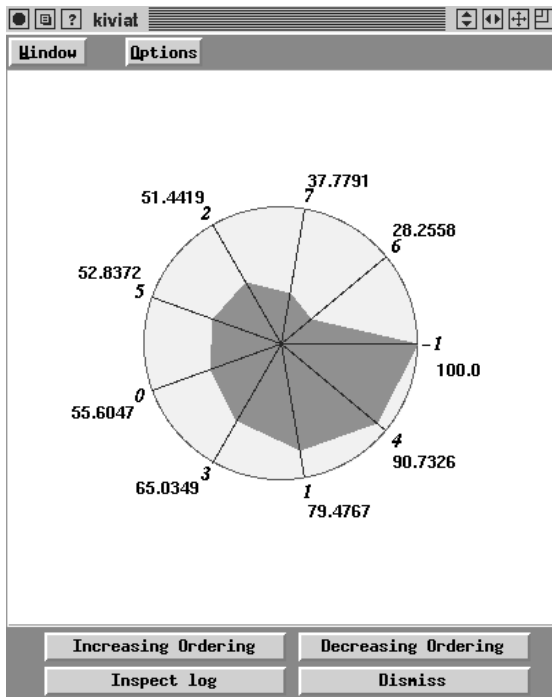


a) Problema B-3

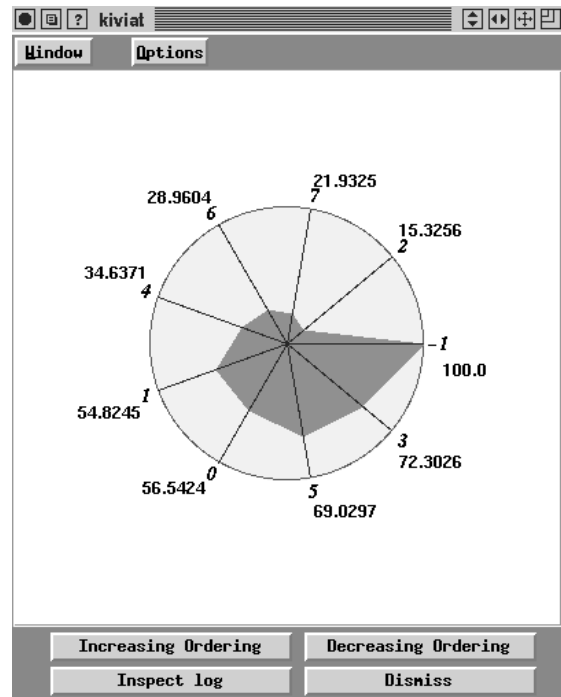


b) Problema B-11

Figura 4.8: Diagramas de Kiviati para a Versão Distribuída de Karp-Zhang



a) Problema B-3



b) Problema B-11

Figura 4.9: Diagramas de Kiviati para a Versão Distribuída de Karp-Zhang Modificada

balanceamento de carga entre os processadores, balanceamento estático, balanceamento de Karp-Zhang e balanceamento de Karp-Zhang modificado, sendo esse último uma proposta original.

Comparando todas as combinações, podemos verificar que elas são competitivas entre si, embora a versão centralizada tenha apresentado o melhor *speedup* para os casos testados. Um resumo dos resultados é ilustrado na Figura 4.4, que apresenta as curvas de *speedup* médio para as quatro implementações. Os resultados dos experimentos conduzidos parecem apontar para um ganho sobre a computação sequencial, em termos de tempo de processamento. Assim, a paralelização de algoritmos *branch-and-bound* é realmente promissora.

Ao longo dos nossos experimentos, observamos que seria desejável que o algoritmo centralizado apresentasse uma melhor taxa de utilização entre os processadores. Esse aspecto parece ser um interessante tópico para estudos futuros. O caso distribuído com balanceamento estático apresenta uma utilização um pouco melhor, mas é fortemente dependente da instância que está sendo resolvida. O uso equilibrado dos processadores é função do balanceamento da árvore de pesquisa da instância. O caso distribuído com balanceamento de Karp-Zhang parece conseguir uma boa taxa de utilização, aliada a uma distribuição de carga mais independente da instância de entrada. As modificações propostas no balanceamento de Karp-Zhang conseguem melhores *speedups* além de manterem essa independência entre distribuição de carga e instância de entrada, mas também ainda não são inteiramente satisfatórias. Acreditamos ter aqui outro tópico para futuras investigações.

Além dessas possíveis extensões, algumas questões permanecem em aberto. Até quanto seria possível manter o *speedup* aproximadamente linear $O(p)$, aumentando-se o número de processadores? Como se comportariam os algoritmos para outras instâncias? Haveria outras formas melhores de paralelizar o algoritmo *branch-and-bound*? Essas são apenas algumas das questões que poderiam ser abordadas em trabalhos futuros.

Capítulo 5

Propostas de Continuidade

Versões sequenciais e paralelas do algoritmo *branch-and-bound* foram desenvolvidas para os problemas NCFCF e PRMN, conforme descrito nos Capítulos 2, 3 e 4. Nesse Capítulo, discutimos possíveis caminhos para uma melhoria dos resultados obtidos até então, com um consequente aumento no tamanho das instâncias tratáveis, dentro de um tempo de processamento razoável. Somos inspirados por trabalhos recentes que tratam tanto subproblemas dos problemas NCFCF e PRMN, quanto problemas similares.

5.1 Heurísticas

Na linha de heurísticas, temos *e.g.* as heurísticas gulosas do tipo *ADD*. Provavelmente, o procedimento *ADD* mais célebre é o algoritmo de Prim para árvore geradora mínima, (Prim, 1957). Heurísticas *ADD* têm sido usadas com sucesso em problemas similares, (Galvão e Raggi, 1989, Hochbaum e Segev, 1989, Cruz *et al.*, 1992b, Mateus *et al.*, 1994).

A procura por heurísticas com garantia teórica de desempenho no pior caso também é uma extensa área de pesquisa. Nessa linha, lembramos os resultados obtidos para problemas de planejamento de topologia de redes, (Balakrishnan *et al.*, 1994c), ou mesmo para o conhecido problema de *Steiner* em grafos, (Klein e Ravi, 1993, Agrawal *et al.*, 1995). Pode ser que também consigamos descobrir para os problemas NCFCF e PRMN heurísticas com garantia de desempenho, ainda que somente para um outro caso particular.

5.2 Limites Inferiores

Considerando o problema de geração de limites inferiores, as relaxações lagrangeanas aplicadas àqueles modelos apresentados nos Capítulos 2 e 3 mostraram-se ser uma boa opção, conforme atestam os nossos resultados computacionais. Entretanto, existem outras alternativas.

Tabela 5.1: Qualidade da Relaxação Linear *versus* Relaxação Lagrangeana

Problema [†]	N	A	D	Relaxação Linear		Relaxação Lagrangeana		
				Limite Inferior	UCP(s) [‡]	Limite Inferior	Limite Superior	UCP(s)
B-1	50	126	8	1.154,25	0,20	1.154,13	1.222*	2,00
2			12	2.450,25	0,27	2.450,24	2.520*	2,20
3			24	4.860,17	0,27	4.859,98	5.012*	2,20
4		200	8	1.174,50	0,32	1.174,49	1.237	4,80
5			12	1.038,58	0,37	1.038,52	1.095*	4,60
6			24	3.072,75	0,38	3.072,51	3.208	5,40
7	75	188	12	2.843,50	0,34	2.843,50	2.943*	4,90
8			18	2.554,11	0,37	2.553,74	2.657*	5,00
9			37	5.655,24	0,36	5.654,49	5.874*	5,10
10		300	12	1.946,08	0,60	1.946,02	2.053	11,00
11			18	3.881,44	0,55	3.881,27	3.987*	11,00
12			37	6.738,16	0,62	6.737,60	6.948	12,00
13	100	250	16	4.266,50	0,43	4.265,95	4.432*	8,90
14			24	8.876,83	0,53	8.876,22	9.117	9,30
15			49	11.052,51	0,54	11.051,60	11.383*	9,20
16		400	16	3.803,63	0,82	3.803,32	3.942	23,00
17			24	5.071,04	0,81	5.070,66	5.193	22,00
18			49	6.092,41	0,94	6.091,92	6.360	21,00

[†] Redes de Beasley, (Beasley, 1990), com $f_{ij}/\Omega_{ij} = 1$ e $c_{ij}/\Omega_{ij} = 10$.

[‡] Tempo de UCP usando o CPLEXTM.

* Soluções ótimas.

5.2.1 Métodos Duais

Uma alternativa é o uso de métodos baseados na dualidade. Métodos duais têm sido usados com sucesso em problemas similares. Resultados computacionais animadores foram apresentados para problemas de p -medianas de grande porte, (Christofides e Beasley, 1982, Galvão e Raggi, 1989), para problemas de *Steiner* em grafos, (Wong, 1984, Maculan, 1987, Beasley, 1989), e problemas de planejamento topológico de redes em multi-níveis, (Balakrishnan *et al.*, 1994a, Balakrishnan *et al.*, 1994c).

5.2.2 Formulações Alternativas

A principal vantagem no emprego da relaxação lagrangeana na resolução dos problemas NCFCF e PRMN é a geração de uma heurística lagrangeana que, conforme demonstrado pelos nossos resultados computacionais, fornece um excelente limite superior para os problemas testados. Os limites inferiores, por outro lado, não são melhores do que aqueles obtidos pela relaxação de programação linear, uma vez que as relaxações lagrangeanas utilizadas atendem à propriedade da integralidade, *i.e.* suas variáveis inteiras possuem soluções ótimas naturalmente inteiras. A igualdade dos limites é uma importante resultado teórico, (Fisher, 1985), facilmente comprovável na prática. Isso é mostrado nos resultados computacionais que apresentamos na Tabela 5.1, onde resolvemos problemas NCFCF, utilizando alguns dos problemas de teste usados no Capítulo 2.

As relaxações lineares foram resolvidas pelo pacote de programação linear CPLEXTM, (CPLEX Optimization, Inc., 1993), usando uma máquina SPARCstation 10, sistema operacional SunOS 5.5, UCP "Model 40 SuperSPARC", e 64 MB de memória RAM. As

relaxações lagrangeanas usadas estão conforme descrito no Capítulo 2. Os limites inferiores das relaxações lagrangeanas foram melhorados pelo algoritmo de otimização por subgradientes, conforme também descrito no Capítulo 2, utilizando a mesma máquina anterior.

Os resultados da Tabela 5.1 mostram a qualidade dos limites inferiores obtidos, nas relaxações linear e lagrangeana, para os problemas na sua forma original, *i.e.* os problemas não sofreram nenhum tipo de redução. Para comparação, também são apresentadas as soluções ótimas respectivas, quando conhecidas, ou senão, o melhor limite superior, obtido por uma heurística lagrangeana.

Ambas as relaxações foram executadas sobre o modelo (M) , apresentado no Capítulo 2 e repetido a seguir:

(M) :

$$\min \sum_{(i,j) \in A} (c_{ij}x_{ij} + f_{ij}y_{ij}), \quad (5.1)$$

s.a.:

$$\sum_{j \in \delta^-(i)} x_{ji} - \sum_{j \in \delta^+(i)} x_{ij} = \begin{cases} -\sum_{k \in D} d_k, & i = s, \\ 0, & \forall i \in T, \\ d_i, & \forall i \in D, \end{cases} \quad (5.2)$$

$$x_{ij} \leq \left(\sum_{k \in D} d_k \right) y_{ij}, \quad \forall (i, j) \in A, \quad (5.3)$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in A, \quad (5.4)$$

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A. \quad (5.5)$$

Conforme já observado anteriormente, (Rardin e Wolsey, 1993, Barahona, 1996), um limite inferior mais forte pode ser obtido usando uma formulação multi-produto para o problema, definida sobre um dígrafo $\mathcal{D} = (N, A)$:

(MP) :

$$\min \sum_{(i,j) \in A} (c_{ij}x_{ij} + f_{ij}y_{ij}), \quad (5.6)$$

s.a.:

$$\sum_{j \in \delta^-(i)} w_{ji}^t - \sum_{j \in \delta^+(i)} w_{ij}^t = \begin{cases} -d_i, & i = s, & \forall t \in D, \\ 0, & \forall i \in N \setminus \{s, t\}, & \forall t \in D, \\ d_i, & i = t, & \forall t \in D, \end{cases} \quad (5.7)$$

$$\sum_{t \in D} w_{ij}^t \leq x_{ij}, \quad \forall (i, j) \in A, \quad (5.8)$$

$$w_{ij}^t \leq d_t y_{ij}, \quad \forall (i, j) \in A, \quad (5.9)$$

$$w_{ij}^t \geq 0, \quad \forall (i, j) \in A, \quad (5.10)$$

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in A. \quad (5.11)$$

Tabela 5.2: Qualidade das Relaxações Lineares para a Razão $f_{ij}/c_{ij} = 0.1$

Problema [†]	N	A	D	Modelo (M)		Modelo (MP)	
				Limite Inferior (%)	UCP(s) [‡]	Limite Inferior (%)	UCP(s) [‡]
B-1	50	126	8	94,46	0,20	100,00	6,40
2			12	97,23	0,27	100,00	50,00
3			24	96,97	0,27	100,00	320,00
4		200	8	94,95	0,32	100,00	190,00
5			12	94,85	0,37	100,00	380,00
6			24	95,78	0,38	100,00	3.500,00
7	75	188	12	96,62	0,34	99,97	190,00
8			18	96,13	0,37	100,00	310,00
9			37	96,28	0,36	*	?
10		300	12	94,79	0,60	100,00	730,00
11			18	97,35	0,55	100,00	3.700,00
12			37	96,98	0,62	*	?
13	100	250	16	96,27	0,43	99,98	650,00
14			24	97,37	0,53	99,99	1.700,00
15			49	97,10	0,54	*	?
16		400	16	96,49	0,82	100,00	6.300,00
17			24	97,65	0,81	*	?
18			49	95,79	0,94	*	?

[†] Redes de Beasley, (Beasley, 1990), com $f_{ij}/\Omega_{ij} = 1$ e $c_{ij}/\Omega_{ij} = 10$.

[‡] Tempo de UCP usando o CPLEXTM.

* Número de restrições > 16.000.

A formulação (MP) consegue fornecer limites inferiores melhores, uma vez que modela mais cuidadosamente os custos fixos nos arcos. Claro que isso é conseguido às custas de um número muito maior de restrições. A Tabela 5.2 compara relaxações de programação linear das duas formulações, (M) e (MP), em termos da qualidade dos limites inferiores gerados, também para os problemas na sua forma original, sem reduções. Os limites inferiores são dados em termos percentuais, relativos às soluções ótimas, quando conhecidas, ou então, ao melhor limite superior conhecido.

Mesmo para os casos apresentados na Tabela 5.2, que têm uma razão f_{ij}/c_{ij} bastante baixa, e onde o modelo (M) tem naturalmente um bom desempenho, a superioridade da formulação (MP) é aparente. Esse desempenho superior é ainda mais pronunciado, se estivermos resolvendo instâncias com razão f_{ij}/c_{ij} alta, onde a formulação (M) apresenta os maiores *gaps* de dualidade. A Tabela 5.3 mostra os resultados obtidos usando uma razão $f_{ij}/c_{ij} = 10$.

Com isso, podemos concluir que uma alternativa promissora para uma resolução mais efetiva dos problemas NCFCF e PRMN é por meio do uso de formulações multi-produto. Entretanto, uma importante questão em aberto é o impacto que o número consideravelmente maior de restrições do modelo (MP) teria (i) sobre o tamanho máximo dos problemas representáveis nos computadores disponíveis e (ii) no tempo de processamento.

5.2.3 Restrições Adicionais

Lembrando os resultados obtidos no Capítulo 2, o cálculo dos limites inferiores para o problema NCFCF se reduz à resolução de um problema linear de fluxos $O(|N| |A|)$, na variável \mathbf{x} , e à resolução do seguinte problema de seleção $O(|A|)$, na variável \mathbf{y} :

Tabela 5.3: Qualidade das Relaxações Lineares para a Razão $f_{ij}c_{ij} = 10$

Problema [†]	N	A	D	Modelo (M)		Modelo (MF)	
				Limite Inferior (%)	UCP(s) [‡]	Limite Inferior (%)	UCP(s) [‡]
B-1	50	126	8	27,46	0,17	100,00	7,60
2			12	41,06	0,18	100,00	59,00
3			24	33,79	0,20	96,11	370,00
4		200	8	35,37	0,25	100,00	180,00
5			12	26,01	0,32	100,00	370,00
6			24	25,17	0,31	93,61	3.700,00
7	75	188	12	36,85	0,26	100,00	210,00
8			18	29,35	0,27	99,48	320,00
9			37	25,34	0,29	*	?
10		300	12	30,96	0,49	95,63	850,00
11			18	40,49	0,41	93,73	3.800,00
12			37	30,87	0,51	*	?
13	100	250	16	32,47	0,37	100,00	730,00
14			24	37,26	0,39	98,19	1.700,00
15			49	28,90	0,43	*	?
16		400	16	35,28	0,73	95,92	6.600,00
17			24	37,52	0,64	*	?
18			49	22,09	0,72	*	?

[†] Redes de Beasley, (Beasley, 1990), com $f_{ij}/\Omega_{ij} = 1$ e $c_{ij}/\Omega_{ij} = 10$.

[‡] Tempo de UCP usando o CPLEXTM.

* Número de restrições > 16.000.

(LR₂):

$$\min \sum_{(i,j) \in A} F_{ij} y_{ij}, \quad (5.12)$$

s.a.:

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in K, \quad (5.13)$$

$$y_{ij} = 1, \quad \forall (i, j) \in K_1, \quad (5.14)$$

$$y_{ij} = 0, \quad \forall (i, j) \in K_0. \quad (5.15)$$

Conforme visto no Capítulo 2, os limites inferiores não são muito justos e uma possibilidade de melhorá-los é pela modificação do problema em \mathbf{y} , acrescentando restrições adicionais, *e.g.*:

(LR'₂):

$$\min \sum_{(i,j) \in A} F_{ij} y_{ij}, \quad (5.16)$$

s.a.:

$$y_{ij} \in \{0, 1\}, \quad \forall (i, j) \in K, \quad (5.17)$$

$$y_{ij} = 1, \quad \forall (i, j) \in K_1, \quad (5.18)$$

$$y_{ij} = 0, \quad \forall (i, j) \in K_0, \quad (5.19)$$

$$|Y| \geq \beta, \quad (5.20)$$

onde $Y = \{(i, j) \mid y_{ij} = 1\}$ e β é o número mínimo de arcos que deverá estar presente na solução, *i.e.* a cardinalidade mínima do conjunto Y .

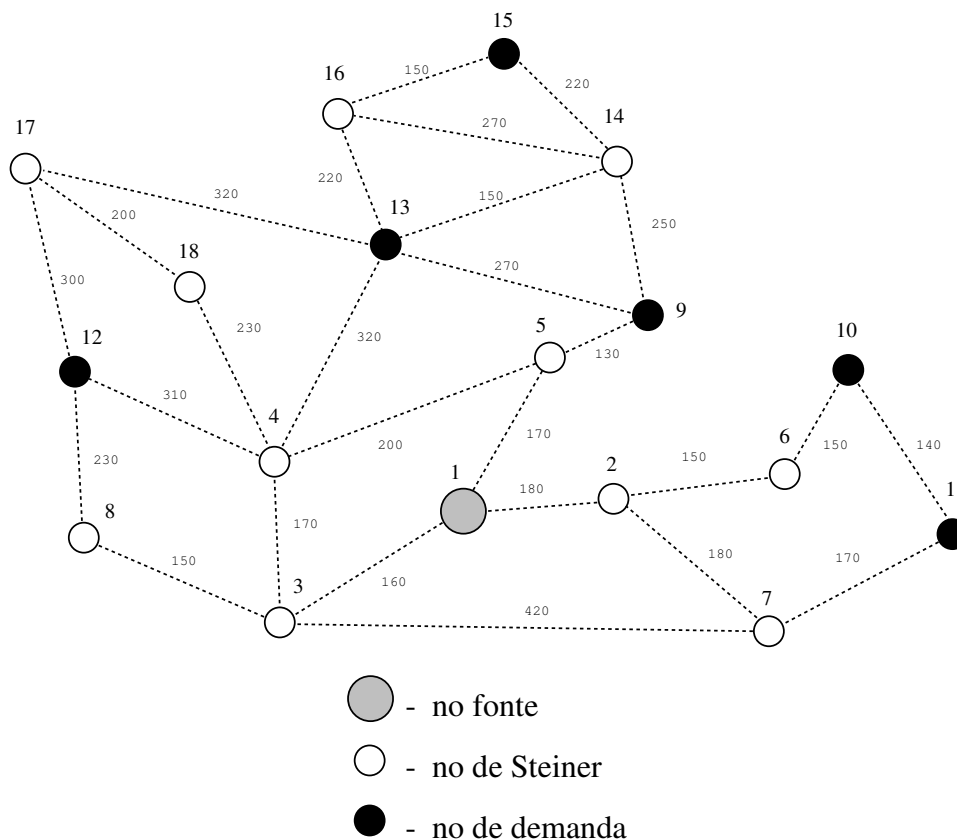


Figura 5.1: Rede Exemplo com $|N| = 18$ e $|A| = 54$

Para *pequenos* valores de β a restrição adicional é redundante e sua inclusão não altera a solução ótima do modelo original. O problema (LR'_2) , embora mais complexo computacionalmente, ainda pode ser resolvido polinomialmente, $O(|A| \log |A|)$. O incremento na complexidade é causado pelo passo de ordenação que precisa ser incluído no algoritmo de resolução.

Um importante questão é se a restrição adicional poderia gerar limites inferiores mais justos e melhorar o desempenho global dos algoritmos *branch-and-bound*. Apresentaremos agora alguns resultados computacionais preliminares comparando as duas abordagens.

Todos os testes aqui apresentados foram executados em uma DECstation 3100 com o sistema operacional ULTRIX V4.2.A. Usamos a rede ilustrada na Figura 5.1. Os números de três dígitos menores indicam os pesos Ω_{ij} das arestas. Os problemas realmente resolvidos foram a versão direcionada dessa rede, com todas as demandas consideradas unitárias. Os custos f_{ij} e c_{ij} foram derivados dos pesos das arestas, segundo os fatores fixos 1 e 10. Os tempos de execução reportados excluem operações de entrada e saída e o tempo gasto com outros processos.

A Tabela 5.4 apresenta os resultados dos experimentos. São apresentados a cardinalidade mínima usada, β , a solução, o *gap*, o número de iterações do algoritmo de subgradi-

Tabela 5.4: Resultados Computacionais para a Rede Exemplo

P#	Modelo	$\frac{c_{ij}}{d_{ij}}$	$\frac{f_{ij}}{d_{ij}}$	β	Primeiro Nó				Árvore de Busca		
					U_{BEST}	GAP(%)	k	UCP(s)	$ Y $	Nós	UCP(s)
1	LR_2	100	1	n/a	321.410*	0,59	313	0,45	13	3	0,10
				0	321.410*	0,59	313	0,68		3	0,16
				6	321.410*	0,46	304	0,65		3	0,16
				9	321.410*	0,32	299	0,65		3	0,16
				13	321.410*	0,13	289	0,63		3	0,17
2	LR_2	10	1	n/a	34.310*	5,50	309	0,46	13	5	0,18
				0	34.310*	5,50	309	0,65		5	0,30
				6	34.310*	4,30	304	0,65		5	0,30
				9	34.310*	3,00	299	0,63		5	0,30
				13	34.310*	1,30	289	0,62		3	0,15
3	LR_2	1	1	n/a	5.600	34,00	312	0,46	12	509	25,00
				0	5.600	34,00	309	0,66		509	42,00
				6	5.600	26,00	304	0,65		697	56,00
				9	5.600	18,00	299	0,64		585	47,00
				12	5.600	11,00	294	0,63		173	13,00
4	LR_2	1	10	n/a	24.050*	64,00	324	0,49	11	2.517	110,00
				0	24.050*	64,00	324	0,70		2.517	190,00
				6	24.050*	48,00	320	0,69		3.865	280,00
				9	24.210	30,00	314	0,67		3.335	240,00
				11	24.210	17,00	309	0,66		1.035	72,00
5	LR_2	1	100	n/a	208.550*	73,00	339	0,52	11	3.587	150,00
				0	208.550*	73,00	339	0,75		3.587	250,00
				6	208.550*	55,00	339	0,76		5.197	380,00
				9	211.410	34,00	330	0,74		4.779	360,00
				11	211.410	19,00	327	0,75		1.425	96,00

*solução ótima

entes, k , e o tempo de UCP em segundos, para o primeiro nó. Também são apresentados o número de arcos na solução ótima, o número de nós explorados na árvore de pesquisa e o tempo de UCP gasto após a primeira exploração. O Problema # 1 usa a razão $c_{ij}/f_{ij} = 100$, aproximando-se de um problema linear de fluxos puro, que é polinomial. Por outro lado, o Problema # 5, usando a razão $c_{ij}/f_{ij} = 1$, aproxima-se de um problema de *Steiner* puro, que é \mathcal{NP} -árduo. Lembramos, entretanto, que todos eles são \mathcal{NP} -árduos.

Para cada um desses problemas, vários casos foram testados. O primeiro utiliza o modelo (LR_2), onde a cardinalidade do conjunto Y é livre. O segundo usa o modelo (LR'_2) e simula a liberação da cardinalidade, estabelecendo o valor 0 para β . Esses dois casos servem apenas para demonstrar na prática o impacto do modelo (LR'_2), no tempo de cálculo. Uma vez que (LR'_2) é computacionalmente mais complexo que (LR_2), os tempos mais longos são compreensíveis.

Os casos seguintes usam apenas o modelo (LR'_2) e fixam a cardinalidade em diferentes valores, até o limite a partir do qual a restrição adicional deixaria de ser redundante. Infelizmente, o cálculo desses limites se reduz exatamente à resolução do problema original.

Um valor mínimo trivial para a cardinalidade de Y é a cardinalidade do conjunto de nós de demanda, $|D|$. Entretanto, esse limite não é bom o bastante, uma vez que em todos os casos que usam $\beta = |D| = 6$ o tempo de processamento foi o maior de todos e o número de nós explorados foi o mais alto. Usando-se o limite mais justo 9, o desempenho foi um pouco melhor, mas não supera o modelo (LR_2). Finalmente, usando o melhor limite possível, o melhor desempenho foi alcançado, principalmente para aqueles problemas próximos ao problema de *Steiner*. Também notamos que o uso de valores mais justos para β pioraram as soluções no primeiro nó, para os Problemas # 4 e # 5.

Assim, algumas interessantes questões surgem. Haveria algum algoritmo polinomial para obter valores de β que fossem justos o bastante para fazer diferença no tempo de processamento? O comportamento aqui observado para a rede da Figura 5.1 seria mantido para outras redes com maior e menor número de nós de demanda, mais esparsas, *etc.*? A investigação dessas questões é uma possível extensão para esse trabalho.

5.2.4 Teoria das Inequações Válidas

Encerrando a seção, cabe lembrar uma conhecida ferramenta em programação inteira, utilizada para melhoria dos limites inferiores, via relaxação linear, a teoria das inequações válidas (*theory of valid inequalities*), (Nemhauser e Wolsey, 1988). Importantes problemas de programação inteira-mista têm sido resolvidos com sucesso por essa técnica, (Bienstock e Günlük, 1995, Barahona, 1996). O uso de relaxações de programação linear acopladas a inequações válidas e cortes podem aumentar ainda mais o tamanho das instâncias tratáveis, por propiciarem limites bem mais justos.

5.3 Testes de Redução

Na linha de testes de redução, foram apresentados eficientes algoritmos de redução para o problema de localização não-capacitado, (Christofides e Beasley, 1982, Mateus e Carvalho, 1992), testes de redução para problemas de localização capacitados, (Beasley, 1988, Mateus e Bornstein, 1991), e para problemas de *Steiner* em grafos, (Maculan *et al.*, 1991). Essa é, portanto, outra possibilidade de tornarmos tratáveis instâncias maiores.

5.4 Conclusões

Apresentamos aqui alguns caminhos promissores para uma melhoria dos nossos resultados. Essas novas possíveis direções de pesquisa são inspiradas por trabalhos recentemente publicados, tratando problemas similares. Parece-nos possível utilizar, com ganhos, qualquer uma das técnicas mencionadas na resolução do problema NCFCF ou mesmo do problema PRMN.

Capítulo 6

Conclusões

Apresentamos nessa tese dois problemas de planejamento de redes com forte apelo teórico e também prático, por generalizarem uma série de outros problemas de otimização em redes conhecidos e extensivamente tratados na literatura da área. Durante a apresentação dos modelos, fizemos uma extensa revisão bibliográfica e acreditamos que uma boa parte dos resultados mais notáveis descobertos na última década tenha sido mencionada.

Apresentamos o problema não-capacitado de fluxos com custos fixos (NCFCF). Mostramos a não-trivialidade na obtenção de uma solução ótima para o problema NCFCF. Usamos um algoritmo exato do tipo *branch-and-bound* para resolvê-lo e propusemos um novo conjunto de procedimentos auxiliares, teoricamente justificados, que reduziram consideravelmente os tempos de processamento. Discutimos possíveis extensões para o estudo.

Introduzimos um novo problema, o qual denominamos problema de planejamento de redes em multi-níveis (PRMN), que é uma extensão do problema NCFCF. Modelamos o problema PRMN e o resolvemos de maneira exata. Para isso, estendemos as idéias desenvolvidas para o problema NCFCF, demonstrando mais uma vez o seu alcance e relevância. Apresentamos algumas questões em aberto e discutimos possíveis extensões para a pesquisa.

Fizemos um estudo empírico comparativo entre várias implementações paralelas do algoritmo *branch-and-bound* aplicado ao problema PRMN. As implementações, adequadas às máquinas paralelas do tipo MIMD, são muito convenientes para processamento em redes de *workstations*, muito populares nos dias de hoje. Os resultados preliminares apontam para direções promissoras. Além disso, durante esse estudo, vários problemas foram levantados, abrindo uma extensa área para futuras investigações.

Baseados nos estudos mais recentemente publicados, mencionamos algumas das possíveis metodologias alternativas para tratamento dos problemas estudados, com vistas à obtenção de resultados ainda melhores.

Finalmente, é importante reforçar que os principais resultados alcançados ao longo dessa pesquisa podem ser estendidos a muitos outros problemas de planejamento de redes.

Apêndice A

Fragmentos de Código em *C*

Em seguida, apresentamos fragmentos na linguagem *C*, para a criação das listas de adjacências e listas encadeadas simples, usadas na implementação dos algoritmos para resolução do problema PRMN.

```

#define Nodes number_of_nodes
#define Levels number_of_levels
/* definition of types */
typedef struct AType *ArcPtrType;
typedef struct AType {
    int          i,j, yij, Status;
    float        cij, Cij, xij, wij,
                Gamma_wij, fij, Fij;
} ArcType;
typedef struct LType *ListArcPtrType;
typedef struct LType {
    ArcPtrType   *APtr;
    ListArcPtrType Next;
} ListArcType;
/* definition of variables */
int             Node, Level;
ArcPtrType     *ArcPtr;
ListArcPtrType *AdjTo, ListArcPtr;
/* initialize adjacency list */
AdjTo = (ListArcPtrType *) calloc( Nodes, sizeof(ListArcPtrType) );
for ( Node = 0; Node < Nodes; Node++) {
    AdjTo[Node] = (ListArcPtrType) malloc( sizeof(ListArcType) );
    AdjTo[Node]->Next = NULL;
}
/* read parameters of all arcs */
while !end_of_arcs {
    ArcPtr = (ArcPtrType *) calloc( Levels, sizeof(ArcPtrType) );
    /* read parameters for all levels */
    for ( Level = 0; Level < Levels; Level++ ) {
        ArcPtr[Level] = (ArcPtrType) malloc( sizeof(ArcType) );
        read_arc_parameter(INFILE, ArcPtr[Level]);
    }
    /* insert arc in list */
    ListArcPtr = (ListArcPtrType) malloc( sizeof(ListArcType) );
    ListArcPtr->APtr = ArcPtr;
    ListArcPtr->Next = AdjTo[index_i_of(ArcPtr)]->Next;
    AdjTo[index_i_of(ArcPtr)]->Next = ListArcPtr;
}

```

Figura A.1: Fragmento de Código em *C* para Construção de Listas de Adjacências

```

#define Levels number_of_levels
/* definition of types */
typedef struct NType *NodPtrType;
typedef struct NType {
    int i, zi, Status;
    float di, fi, Fi, vi, Gamma_vi, Sigma_i, Flow_Cr;
} NodType;
typedef struct LNTType *ListNodPtrType;
typedef struct LNTType {
    NodPtrType      NPtr;
    ListNodPtrType Next;
} ListNodType;

/* definition of variables */
int      Node, Level;
NodPtrType      NodPtr;
ListNodPtrType  *Roots, *Demands, ListNodPtr;
/* initialize singly-linked lists */
Roots = (ListNodPtrType *) calloc(Levels, sizeof(ListNodPtrType));
Demands = (ListNodPtrType *) calloc(Levels, sizeof(ListNodPtrType));
for ( Level = 0; Level < Levels; Level++) {
    Roots[Level] = (ListNodPtrType) malloc( sizeof(ListNodType) );
    Roots[Level]->Next = NULL;
    Demands[Level] = (ListNodPtrType) malloc( sizeof(ListNodType) );
    Demands[Level]->Next = NULL;
}
/* read parameters of all candidate supply nodes */
while !end_of_supply_nodes {
    /* read and insert node in list */
    read_supply_node_parameters(INFILE, NodPtr);
    ListNodPtr = (ListNodPtrType) malloc( sizeof(ListNodType) );
    ListNodPtr->NPtr = NodPtr;
    ListNodPtr->Next = Roots[level_of(NodPtr)]->Next;
    Roots[level_of(NodPtr)]->Next = ListNodPtr;
}
/* read parameters of all demand nodes */
while !end_of_demand_nodes {
    /* read and insert node in list */
    read_demand_node_parameters(INFILE, NodPtr);
    ListNodPtr = (ListNodPtrType) malloc( sizeof(ListNodType) );
    ListNodPtr->NPtr = NodPtr;
    ListNodPtr->Next = Demands[level_of(NodPtr)]->Next;
    Demands[level_of(NodPtr)]->Next = ListNodPtr;
}

```

Figura A.2: Fragmento de Código em *C* para Construção de Listas Encadeadas Simples

Bibliografia

- [Aarts e Korst, 1989] E. Aarts e J. Korst. *Simulated Annealing & Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, New York, 1989.
- [Agrawal *et al.*, 1995] A. Agrawal, P. Klein e R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing*, 24(3):440–456, 1995.
- [Aho *et al.*, 1983] A.V. Aho, J.E. Hopcroft e J.D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, Massachusetts, 1983.
- [Altinkemer e Gavish, 1988] K. Altinkemer e B. Gavish. Heuristics with constant error guarantees for the design of tree networks. *Management Science*, 32:331–341, 1988.
- [Aneja, 1980] Y.P. Aneja. An integer linear programming approach to Steiner problem in graphs. *Networks*, 10:167–178, 1980.
- [Balakrishnan e Altinkemer, 1992] A. Balakrishnan e K. Altinkemer. Using a hop-constrained model to generate alternative communication network design. *ORSA Journal on Computing*, 4:192–205, 1992.
- [Balakrishnan *et al.*, 1991] A. Balakrishnan, T.L. Magnanti, A. Shulman e R.T. Wong. Models for planning capacity expansion in local access telecommunication networks. *Annals of Operations Research*, 33:239–284, 1991.
- [Balakrishnan *et al.*, 1994a] A. Balakrishnan, T.L. Magnanti e P. Mirchandani. A dual-based algorithm for multi-level network design. *Management Science*, 40(7):567–581, 1994.
- [Balakrishnan *et al.*, 1994b] A. Balakrishnan, T.L. Magnanti e P. Mirchandani. Designing hierarchical survivable networks. Working Paper OR 291-94, Sloan School of Management, MIT, 1994.
- [Balakrishnan *et al.*, 1994c] A. Balakrishnan, T.L. Magnanti e P. Mirchandani. Modeling and heuristic worst-case performance analysis of two-level network design problem. *Management Science*, 40(7):846–867, 1994.

- [Barahona, 1996] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6(3):823–837, 1996.
- [Barr *et al.*, 1981] R.S. Barr, F. Glover e D. Klingman. A new optimization method for large scale fixed charge transportation problems. *Operations Research*, 29:448–463, 1981.
- [Barrera *et al.*, 1993] T. Barrera, J. Griffith, G. Robins e T. Zhang. Closing the gap: Near-optimal Steiner trees in polynomial time. Relatório Técnico CS-93-31, Department of Computer Science, University of Virginia, Charlottesville, EUA, 1993.
- [Bazaraa *et al.*, 1990] M.S. Bazaraa, J.J. Jarvis e H.D. Sherali. *Linear Programming and Networks Flows*. John Wiley & Sons, New York, 2nd edition, 1990.
- [Beasley, 1988] J.E. Beasley. An algorithm for solving large capacitated warehouse location problems. *Journal of the Operational Research Society*, 33:314–325, 1988.
- [Beasley, 1989] J.E. Beasley. An SST-based algorithm for the Steiner problem in graphs. *Networks*, 19:1–16, 1989.
- [Beasley, 1990] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.
- [Bienstock e Günlük, 1995] D. Bienstock e O. Günlük. Computational experience with a difficult mixed-integer multicommodity flow problem. *Mathematical Programming*, 68:213–237, 1995.
- [Cabot e Erenguc, 1984] A.V. Cabot e S.S. Erenguc. Some branch-and-bound procedures for fixed-cost transportation problems. *Naval Research Logistics Quarterly*, 31:145–154, 1984.
- [Christofides e Beasley, 1982] N. Christofides e J.E. Beasley. A tree search algorithm for the p-median problem. *European Journal of Operational Research*, 10:196–204, 1982.
- [CPLEX Optimization, Inc., 1993] CPLEX Optimization, Inc. *Using the CPLEX™ Callable Library and CPLEX™ Mixed Integer Library*. Incline Village, NV, 1993.
- [Cruz e Mateus, 1997] F.R.B. Cruz e G.R. Mateus. Parallel implementations of branch-and-bound algorithms applied to a network design problem. Relatório Técnico RT002/97, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brasil, 1997. (a ser apresentado na *EURO/INFORMS 1997 Joint International Meeting*).
- [Cruz *et al.*, 1991] F.R.B. Cruz, G.R. Mateus e H.P.L. Luna. Algorithm for hierarchical network design. In *TIMS/ORSA Joint National Meeting*, p. 64, Nashville, EUA, 1991.
- [Cruz *et al.*, 1992a] F.R.B. Cruz, H.P.L. Luna e G.R. Mateus. Aplicação do problema das p-medianas com custos fixos ao projeto de redes hierárquicas. In R.C. Souza, editor, *Anais do XXIV Simpósio Brasileiro de Pesquisa Operacional*, p. 124–128, Salvador, Brasil, 1992. SOBRAPO.

- [Cruz *et al.*, 1992b] F.R.B. Cruz, H.P.L. Luna e G.R. Mateus. Uma heurística para o problema de planejamento de redes telefônicas de alimentação. In E.P. Ferreira, editor, *Anais do 9^o Congresso Brasileiro de Automática*, p. 443–448, Vitória, Brasil, 1992. SBA.
- [Cruz *et al.*, 1994] F.R.B. Cruz, J.A. Almeida e G.R. Mateus. Análise do problema de planejamento de redes telefônicas de alimentação. In M.A. Silveira and P.M.G. Ferreira, editors, *Anais do 10^o Congresso Brasileiro de Automática*, volume 1, p. 572–577, Rio de Janeiro, Brasil, 1994. SBA.
- [Cruz *et al.*, 1995a] F.R.B. Cruz, J. MacGregor Smith e G.R. Mateus. A branch-and-bound algorithm to solve the multi-level network optimization problem. Relatório Técnico RT030/95, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brasil, 1995. (submetido à revista *Networks*).
- [Cruz *et al.*, 1995b] F.R.B. Cruz, J. MacGregor Smith e G.R. Mateus. Solving to optimality the uncapacitated fixed-charge network flow problem. Relatório Técnico RT031/95, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brasil, 1995. (em revisão com a revista *Computers & Operations Research*).
- [Cruz *et al.*, 1996a] F.R.B. Cruz, G.R. Mateus e J. MacGregor Smith. Algorithm to solve the multi-level network optimization problem. In *Fifth SIAM Conference on Optimization*, p. 80, Victoria, Canada, 1996.
- [Cruz *et al.*, 1996b] F.R.B. Cruz, G.R. Mateus e J. MacGregor Smith. A branching strategy for uncapacitated fixed-charge network flow problems. In *Anais do 11^o Congresso Brasileiro de Automática*, p. 359–364, São Paulo, Brasil, 1996. SBA.
- [Cruz *et al.*, 1996c] F.R.B. Cruz, G.R. Mateus e J. MacGregor Smith. The multi-level network optimization problem. In *Anais do 11^o Congresso Brasileiro de Automática*, p. 365–370, São Paulo, Brasil, 1996. SBA.
- [Cruz *et al.*, 1996d] F.R.B. Cruz, G.R. Mateus e J. MacGregor Smith. A reduction test based on Lagrangean relaxation applied to uncapacitated fixed-charge network flow problems. In *Proceedings of the VIII Latin-Iberian-American Congress on Operations Research and System Engineering*, p. 519–524, Rio de Janeiro, Brasil, 1996. SOBRAPO.
- [Cruz *et al.*, 1996e] F.R.B. Cruz, J. MacGregor Smith e G.R. Mateus. Algorithms for the multi-level network optimization problem. Relatório Técnico RT007/96, Departamento de Ciência da Computação, UFMG, Belo Horizonte, Brasil, 1996. (em revisão com a revista *European Journal of Operational Research*).
- [Cruz, 1991] F.R.B. Cruz. Um algoritmo para projeto de redes hierárquicas. Dissertação de Mestrado, DCC-ICEX-UFMG, Belo Horizonte, MG, 1991.
- [Current *et al.*, 1986] J.R. Current, C.S. ReVelle e J.L. Cohon. The hierarchical network design problem. *European Journal of Operational Research*, 27:57–66, 1986.

- [Duin e Volgenant, 1989a] C.W. Duin e A. Volgenant. Reducing the hierarchical network design problem. *European Journal of Operational Research*, 39:332–344, 1989.
- [Duin e Volgenant, 1989b] C.W. Duin e A. Volgenant. Reduction tests for the Steiner problem in graphs. *Networks*, 19:549–567, 1989.
- [Erlenkotter, 1978] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.
- [Fisher, 1985] M.L. Fisher. An application oriented guide to Lagrangean relaxation. *Interfaces*, 15:10–21, 1985.
- [Galvão e Raggi, 1989] R.D. Galvão e L.A. Raggi. A method for solving to optimality uncapacitated location problems. *Annals of Operations Research*, 18:225–244, 1989.
- [Galvão, 1993] R.D. Galvão. The use of Lagrangean relaxation in the solution of uncapacitated facility location problems. *Location Science*, 1(1):57–79, 1993.
- [Garey e Johnson, 1979] M.R. Garey e D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
- [Gavish, 1982] B. Gavish. Topological design of centralized computer networks - Formulations and algorithms. *Networks*, 12:355–377, 1982.
- [Gavish, 1991] B. Gavish. Topological design of telecommunication networks - Local access design methods. *Annals of Operations Research*, 33:17–71, 1991.
- [Gavish, 1992] B. Gavish. Topological design of computer communication networks - The overall design problem. *European Journal of Operational Research*, 58:149–172, 1992.
- [Gendreau *et al.*, 1995] M. Gendreau, M. Labbé e G. Laporte. Efficient heuristics for the design of ring networks. *Telecommunication Systems*, 4(3-4):177–188, 1995.
- [Geoffrion e McBride, 1978] A.M. Geoffrion e R. McBride. Lagrangean relaxation applied to capacitated facility location problems. *AIIE Transactions*, 10:40–47, 1978.
- [Glover *et al.*, 1985] F. Glover, D. Klingman e N. Phillips. A new polynomially bounded shortest path algorithm. *Operations Research*, 33(1):65–73, 1985.
- [Glover, 1989] F. Glover. Tabu search - Part I. *ORSA Journal on Computing*, 1:190–206, 1989.
- [Glover, 1990] F. Glover. Tabu search - Part II. *ORSA Journal on Computing*, 2:4–32, 1990.
- [Goemans e Myung, 1993] M.X. Goemans e Y. Myung. A catalog of Steiner tree formulations. *Networks*, 23:19–28, 1993.

- [Goemans *et al.*, 1994] M.X. Goemans, A.V. Goldeberg, S. Plotkin, D. Shmoys, É. Tardos e D.P. Williamson. Improved approximation algorithms for network design problems. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, p. 223–232, 1994.
- [Goemans, 1994] M.X. Goemans. The Steiner tree polytope and related polyhedra. *Mathematical Programming*, 63:157–182, 1994.
- [Held *et al.*, 1974] M. Held, P. Wolfe e H.D. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.
- [Hochbaum e Segev, 1989] D.S. Hochbaum e A. Segev. Analysis of a flow problem with fixed charges. *Networks*, 19:291–312, 1989.
- [Karp e Zhang, 1993] R.M. Karp e Y. Zhang. Randomized parallel algorithms for backtrack search and branch-and-bound computation. *Journal of the ACM*, 40(3):765–789, 1993.
- [Klein e Ravi, 1993] P.N. Klein e R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. In *Proc. 3rd Symposium on Integer Programming and Combinatorial Optimization*, p. 323–332, 1993.
- [Laursen, 1993] P.S. Laursen. Simple approaches to parallel branch-and-bound. *Parallel Computing*, 19:143–152, 1993.
- [Luna *et al.*, 1987] H.P.L. Luna, N. Ziviani e R.M.B. Cabral. The telephonic switching centre network problem: Formalization and computational experience. *Discrete Applied Mathematics*, 18:199–210, 1987.
- [Maculan *et al.*, 1991] N. Maculan, P. Souza e A.C. Vejar. An approach for the Steiner problem in directed graphs. *Annals of Operations Research*, 33:471–480, 1991.
- [Maculan, 1987] N. Maculan. The Steiner problem in graphs. *Annals of Discrete Mathematics*, 31:185–212, 1987.
- [Magnanti *et al.*, 1986] T.L. Magnanti, P. Mireault e R.T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 26:112–154, 1986.
- [Mateus e Bornstein, 1991] G.R. Mateus e C.T. Bornstein. Dominance criteria for the capacitated warehouse location problem. *Journal of the Operational Research Society*, 42:145–149, 1991.
- [Mateus e Carvalho, 1992] G.R. Mateus e J.C.P. Carvalho. O problema de localização não capacitado: Modelos e algoritmos. *Investigación Operativa*, 2:297–317, 1992.
- [Mateus e Luna, 1989] G.R. Mateus e H.P.L. Luna. Combinatorial optimization in telephonic network planning. In *Workshop on Practical Combinatorial Optimization*, p. 40–54, Rio de Janeiro, Brasil, 1989. IFORS/ALIO.

- [Mateus *et al.*, 1994] G.R. Mateus, F.R.B. Cruz e H.P.L. Luna. An algorithm for hierarchical network design. *Location Science*, 2(3):149–164, 1994.
- [Nemhauser e Wolsey, 1988] G.L. Nemhauser e L.A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988.
- [Papadimitriou e Steiglitz, 1982] C.H. Papadimitriou e K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, New York, 1982.
- [Prim, 1957] R.C. Prim. Shortest connection networks and some generalizations. *Bell System Techn. J.*, 36:1389, 1957.
- [Quinn, 1987] M.J. Quinn. *Designing Efficient Algorithms for Parallel Computers*. McGraw-Hill Book Company, New York, first edition, 1987.
- [Rardin e Wolsey, 1993] R. L. Rardin e L. A. Wolsey. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research*, 71:95–109, 1993.
- [Rothfarb *et al.*, 1970] B. Rothfarb, H. Frank, D.M. Rosembaum e K. Steiglitz. Optimal design of offshore natural-gas pipeline systems. *Operations Research*, 18:992–1020, 1970.
- [Santos e Galvão, 1992] H.N. Santos e R.D. Galvão. Uma escolha para o tamanho do passo em um método de subgradientes aplicado ao problema de localização capacitado. *Pesquisa Operacional*, 12(1):47–61, 1992.
- [Suhl, 1985] U. Suhl. Solving large scale mixed integer programs with fixed charge variables. *Mathematical Programming*, 32:165–182, 1985.
- [Tanenbaum, 1981] A.S. Tanenbaum. *Computer Networks*. Prentice-Hall, New York, 1981.
- [Tarjan, 1983] R.E. Tarjan. *Data Structures and Network Algorithms*. SIAM Publications, 1983.
- [Tavares *et al.*, 1995] A.I. Tavares, M.L.B. Carvalho, and G.R. Mateus. Aided design and analysis of distributed branch-and-bound algorithms. In *Annals of XV International Conference of the Chilean Society of Computer Science*, p. 448–458, Arica, Chile, 1995. Chilean Society of Computer Science.
- [Tavares, 1995] A.I. Tavares. Um sistema para implementação e análise de técnicas de *branch-and-bound* em redes de estações de trabalho. Dissertação de Mestrado, DCC-ICEx-UFMG, Belo Horizonte, MG, 1995.
- [Wong, 1984] R.T. Wong. A dual ascent algorithm for the Steiner problem in directed graphs. *Mathematical Programming*, 28:271–287, 1984.
- [Ziviani, 1993] N. Ziviani. *Projeto de Algoritmos - Com Implementações em Pascal e C*. Pioneira, São Paulo, Brasil, 1993.