MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

MÉTODOS DE GERAÇÃO DE COLUNAS PARA PROBLEMAS DE ATRIBUIÇÃO

Silvely Nogueira de Almeida Salomão

Tese de Doutorado em Computação Aplicada, orientada pelo Dr. Luiz Antonio Nogueira Lorena e pelo Prof. Dr. Edson Luiz França Senne.

INPE
São José dos Campos
2005

519.8

SALOMÃO, S.N.A.

Métodos de Geração de Colunas para Problemas de Atribuição / S. N. A. Salomão – São José dos Campos: INPE, 2005.

146p. – ().

1. Otimização Combinatória. 2. Problemas de Atribuição. 3. Problema Generalizado de Atribuição. 4. Relaxação Lagrangeana/*Surrogate*. 5. Geração de Colunas. 6. *Branch-and-Price*.

1. Combinatorial Optimization. 2. Assignment Problems. 3. Generalized Assignment Problem. 4. Langrangean/Surrogate Relaxation. 5. Column Generation. 6. Branch-and-Price.

FOLHA DE APROVAÇÃO

"Não sejais demasiadamente justo, nem exageradamente sábio;
por que destruirias a ti mesmo".

(Provérbio de Salomão)

A meus pais,
LEÃO SALOMÃO e
SYLVIA NOGUEIRA DE ALMEIDA SALOMÃO.

AGRADECIMENTOS

Agradeço a Deus em primeiro lugar, pois tenho certeza que foi Ele quem permitiu que eu fizesse este curso de doutorado.

À Universidade Estadual Paulista – UNESP, pela dispensa de 3 anos em período integral de minhas atividades docentes e de 2,5 anos em período parcial para que pudesse concluir esta pesquisa.

À Fundação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, pelo auxilio financeiro através de bolsa PICDT de doutorado por três anos.

Ao Instituto Nacional de Pesquisas Espaciais – INPE, pela oportunidade de estudar num centro de pesquisa tão importante e pela qualidade tão esmeradamente mantida.

Aos amigos do INPE Sérgio Donizete Farias, Juliana Braga, Adriana e Arley, pela amizade que sempre tivemos e ainda prevalece.

Aos colegas de curso Lamosa, Ana, Élcio, Pin (Fabrício), Léo, Manoel, Ana, Alexandre, Patrícia, e a todos com quem pude aprender um pouco mais da computação aplicada.

Aos professores do INPE pelo conhecimento compartilhado.

Aos orientadores Dr. Luiz Antonio Nogueira Lorena e Prof. Dr. Edson Luiz França Senne, pelo tempo gasto comigo, pelo conhecimento passado, e pela orientação e apoio na realização deste trabalho.

Agradeço especialmente às pessoas mais próximas à mim: minha família e ao Tum (José Rubens) por ter suportado meu mau humor, minhas ansiedades que muitas vezes aparecerem devido às minhas angústias provenientes do doutorado.

RESUMO

Este trabalho propõe métodos de geração de colunas para dois importantes problemas de atribuição: o Problema Generalizado de Atribuição (PGA) e o Problema de Atribuição de Antenas Celulares a Comutadores (PAAC). PGA é um dos mais representativos problemas de Otimização Combinatória e consiste em otimizar a atribuição de n tarefas a m agentes, de forma que cada tarefa é atribuída a exatamente um agente e a capacidade de cada agente seja respeitada. PAAC consiste em determinar qual a maneira ótima de atribuir m comutadores a n antenas com posições fixas em uma dada região, de forma a minimizar todos os custos envolvidos, que compreendem custos de cabeamento entre antenas e comutadores e custos de transferência de chamadas entre comutadores. Os dois problemas são conhecidos serem NP-difíceis. A abordagem tradicional de geração de colunas é comparada com a proposta neste trabalho, que utiliza a relaxação lagrangeana/*surrogate*. O trabalho propõe também um método *branch-and-price* para o Problema Generalizado de Atribuição, que utiliza a nova abordagem de geração de colunas proposta. São apresentados testes computacionais que demonstram a efetividade dos algoritmos propostos.

COLUMN GENERATION METHODS FOR ASSIGNMENT PROBLEMS

ABSTRACT

This work proposes column generation methods for two important assignment problems: the Generalized Assignment Problem (GAP) and the problem of assigning cells to switches in cellular mobile networks (PACS). GAP is one of the most representative combinatorial optimisation problem and can be stated as the problem of optimising the assignment of n jobs to m agents, such that each job is assigned to exactly one agent and the resource capacity of each agent is not violated. PACS consists of determining a cell assignment pattern which minimizes a cost function while respecting certain constraints, especially those related to limited switch's capacity. The costs involved correspond to the cabling costs between a cell and a switch and transfer costs between cells assigned to different switches. Both are known to be NP-hard problems. The traditional column generation process is compared with the proposed algorithm that combines the column generation and lagrangean/surrogate relaxation. A branch-and-price method for the GAP which uses the new column generation approach is also proposed in this work. Computational experiments are presented in order to confirm the effectiveness of the proposed algorithms.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$|x|$ – Valor absoluto de x

$\Re$ – Conjunto dos números reais

$\vee$ – Operação “ou”

LISTA DE SIGLAS E ABREVIATURAS

AG	–	Algoritmo Genético
B&B	–	*Branch-and-Bound*
B&P	–	*Branch-and-Price*
BT	–	Busca Tabu
GC	–	Geração de Colunas
NP	–	Polinomial Não-determinístico
PAACC	–	Problema de Atribuição de Antenas Celulares a Comutadores
PAM	–	Problema de Atribuição Multidimensional
PCC	–	Problema de Cobertura de Conjuntos
PGA	–	Problema Generalizado de Atribuição
PL	–	Programação Linear
PLA	–	Problema Linear de Atribuição
PMR	–	Problema Mestre Restrito
PQA	–	Problema Quadrático de Atribuição
SA	–	*Simulated Annealing*

CAPÍTULO 1

PROBLEMAS DE ATRIBUIÇÃO

O problema de atribuição é um problema clássico de Otimização Combinatória em Pesquisa Operacional. De uma forma geral, o problema pode ser estabelecido como: um conjunto de m agentes deve ser atribuído a um conjunto de n tarefas com um custo de atribuição associado. É necessário executar todas as tarefas, atribuindo-se a cada uma delas apenas um agente, de tal forma que o custo total da atribuição seja minimizado. Este problema tem grande importância prática, estando presente, por exemplo, no simples planejamento diário das tarefas de uma pessoa, em que as tarefas devem ser distribuídas de modo a aproveitar melhor o tempo. No entanto, problemas de atribuição aparecem em problemas importantes e bem mais complexos como, por exemplo, a distribuição de tarefas entre processadores de um sistema de computação paralela (Bokhari, 1987), o planejamento de tarefas do telescópio espacial ROSAT (Nowakovski *et al.*, 1999), distribuição de pacientes em vôos médicos (Ruland, 1999), a atribuição de freqüências em redes de comunicação celular (Fischetti *et al.*, 2000; Wang e Gu, 2004), dentre inúmeros outros. Além disto, existem muitos problemas de decisão que, caso não sejam diretamente um problema de atribuição, contêm um problema de atribuição como um subproblema.

De uma forma geral, dados dois conjuntos discretos I e J, pode-se entender uma atribuição de elementos $i \in I$ a elementos $j \in J$ como uma função $x: I \times J \rightarrow [0,1]$. Dessa forma, um problema de atribuição consiste em determinar uma atribuição factível $x \in X$ que minimize uma dada função objetivo f(x). Assumindo que f(x) é decomponível, isto é, $f(x) = \sum_{j \in J} f_j(x)$ e $X = \bigcup_{j \in J} X_j$, o problema pode ser formulado como:

$$\min \quad f(x) = \sum_{j \in J} f_j(x) \tag{1.1}$$

sujeito a:

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \tag{1.2}$$

$$x_j \in X_j \subseteq [0,1]^{|I|}, \quad \forall j \in J \tag{1.3}$$

Problemas de atribuição envolvem, em geral, a indivisibilidade. Assim, se os elementos a serem atribuídos correspondem a atividades, por exemplo, estas devem ser atribuídas a exatamente um outro elemento (um recurso, por exemplo). A indivisibilidade em problemas de atribuição leva a problemas de Otimização Combinatória difíceis. Essa característica torna os problemas de atribuição interessantes do ponto de vista de pesquisa, pois encontrar algoritmos de resolução que resultem em soluções eficientes (boa qualidade e com o mínimo de tempo computacional) para tal classe de problemas constitui um grande desafio.

Existem diversos tipos de problemas de atribuição. Em sua forma mais simples, o problema não envolve restrições de capacidade. Neste caso, quando m = n, ou seja, quando o número de agentes é igual ao número de tarefas, o problema é conhecido como Problema Linear de Atribuição (PLA). Neste caso, o problema é um caso especial de outro problema de otimização, conhecido como Problema de Transporte. Devido à sua estrutura especial, o PLA pode ser resolvido eficientemente (em tempo polinomial) por diversos métodos bem conhecidos (Kuhn, 1955; Hung e Rom, 1980; Wright, 1990).

Outro tipo de problema de atribuição é conhecido como Problema Quadrático de Atribuição (PQA). O problema ocorre quando, para designar objetos a locais, deve-se levar em conta as distâncias entre os pares de locais e os fluxos de algum tipo de demanda entre os pares de objetos. O PQA consiste em encontrar uma alocação de custo mínimo dos objetos aos locais, sendo os custos obtidos por produtos distância $\times$ fluxo. O primeiro trabalho sobre este tipo de problema de atribuição foi publicado por Koopmans e Beckmann (1957) e, desde então, têm sido inúmeras as aplicações do problema para representar modelos logísticos, econômicos e de planejamento. O problema é conhecido ser da classe NP-difícil e aparece em diversas situações práticas, como em projeto de circuitos eletrônicos, em problemas de escalonamento de horário,

em análise estatística, em planejamentos de hospitais, dentre muitos outros. Uma extensa revisão sobre o PQA pode ser vista em Loiola *et al.* (2004).

Outro tipo de problema ocorre quando elementos de um número variável de conjuntos devem ser atribuídos mutuamente. Este problema é conhecido como Problema de Atribuição Multidimensional (PAM) (Pierskalla, 1968) e pode ser estabelecido da seguinte forma (Grundel *et al.*, 2005): Sejam $A_1, ..., A_d$, d conjuntos de elementos (d é conhecido como a dimensão do problema) e seja n_i, o número de elementos do conjunto A_i (i = 1, ..., d). Seja $x_{i_1 ... i_d}$, uma variável binária que estabelece se o elemento i_1 de A_1 está atribuído ao elemento i_j do conjunto A_j (j = 2, ..., d). Seja $c_{i_1 ... i_d}$, o custo da atribuição ($i_1, ..., i_d$). Então o PAM pode ser estabelecido como:

$$\min \quad \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} c_{i_1 \ldots i_d} x_{i_1 \ldots i_d} \tag{1.4}$$

sujeito a:

$$\sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} x_{i_1 \ldots i_d} = 1 \quad \forall\, i_1 = 1, ..., n_1 \tag{1.5}$$

$$\sum_{i_1=1}^{n_1} \cdots \sum_{i_{k-1}=1}^{n_{k-1}} \sum_{i_{k+1}=1}^{n_{k+1}} \cdots \sum_{i_d=1}^{n_d} x_{i_1 \ldots i_d} \le 1 \quad \forall\, k = 2, ..., d-1 \quad \text{e} \quad i_k = 1, ..., n_k \tag{1.6}$$

$$\sum_{i_1=1}^{n_1} \cdots \sum_{i_{d-1}=1}^{n_{d-1}} x_{i_1 \ldots i_d} \le 1 \quad \forall\, i_d = 1, ..., n_d \tag{1.7}$$

$$x_{i_1 \ldots i_d} \in \{0,1\} \quad \forall\, i_1, ..., i_d \in \{1, ..., n\}, \; n_1 \le n_2 \le \cdots \le n_d \tag{1.8}$$

Diversas situações relatadas na literatura envolvem o PAM, dentre as quais, a determinação de trajetórias de partículas elementares (Pusztaszeri *et al.*, 1996) e a determinação de células a partir de uma seqüência de imagens (Kirubarajan *et al.*, 2001).

Existem várias outras variações do problema de atribuição, como o *Problema de Atribuição Dinâmica*, onde a atribuição de agentes a tarefas deve ser feita de forma dinâmica sobre períodos de tempo, o que é muito comum em problemas de roteamento e escalonamento, o *Problema de Atribuição Aleatória*, que corresponde em encontrar em uma matriz $m \times n$ de variáveis aleatórias, k elementos (sendo que não existem dois elementos na mesma linha ou coluna), tais que sua soma seja mínima, o *Problema de Atribuição Axial*, que consiste em encontrar um clique de peso mínimo para um grafo tripartido completo, dentre muitas outras. Uma revisão sobre os principais problemas de atribuição pode ser encontrada em Burkard (2002).

Neste trabalho, pretende-se estudar e propor métodos de resolução para duas formas do problema de atribuição para as quais, existem restrições de capacidade e o número de agentes é menor do que o número de tarefas ($m < n$): o *Problema Generalizado de Atribuição* (PGA), um problema clássico da literatura, e o *Problema de Atribuição de Antenas Celulares a Comutadores* (PAACC), um problema mais recente e que ganhou importância com o surgimento das redes de telefonia celular. Estes dois tipos de problemas de atribuição, tanto o PGA como o PAACC, fazem parte da classe de problemas combinatórios NP-difíceis (Garey e Johnson, 1979).

O PGA pode ser visto como o problema de atribuir n tarefas a m máquinas ao menor custo possível, de modo que cada tarefa seja atribuída a apenas uma única máquina e cada máquina por sua vez, não exceda sua capacidade máxima de trabalho.

Sejam:

- c_{ij} o custo associado de atribuir a tarefa j à máquina i;
- r_{ij} um inteiro positivo que representa o tempo que a máquina i leva para realizar a tarefa j, se a máquina i estiver alocada para a tarefa j;
- b_i um inteiro positivo que representa a capacidade total da máquina i;
- x_{ij} uma variável definida por:

$$x_{ij} = \begin{cases} 1 \text{ se a tarefa } j \text{ é designada à máquina } i \\ 0 \text{ caso contrário} \end{cases} \quad (1.9)$$

Com isto, a formulação do Problema Generalizado de Atribuição pode ser dada por:

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \quad (1.10)$$

sujeito a:

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \ldots, n \quad (1.11)$$

$$\sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i \qquad i = 1, \ldots, m \quad (1.12)$$

$$x_{ij} \in \{0,1\} \qquad i = 1, \ldots, m; \; j = 1, \ldots, n \quad (1.13)$$

A restrição (1.11) elimina a possibilidade da tarefa j ser atribuída a mais de uma máquina e a restrição (1.12) elimina a possibilidade da máquina i ser atribuída a mais tarefas que ela possa executar.

Uma das primeiras aplicações do PGA se deve a Balachandran (1976), que estudou o problema de atribuir tarefas em uma rede de computadores. Desde então, muitos outros problemas têm sido apresentados na literatura como aplicações do PGA, dentre os quais podem ser citados:

- Problema de alocação de salas de aula (Luan e Yao, 1996);
- Problema de roteamento de veículos (Baker e Sheasby, 1999);
- Problema de alocação de pacientes em vôos médicos (Ruland, 1999);
- Problema de carregamento de caminhões (Pigatti, 2003) e

- Problema de recuperação de blocos de dados a partir de discos paralelos (Aerts *et al*., 2003).

Mesmo o PGA apresenta várias formas. Uma forma bem conhecida é o *Problema Generalizado de Atribuição Multi-nível* (Laguna *et al*. 1995; French e Wilson, 2002; Osorio e Laguna, 2003), em que, na atribuição de agentes a tarefas, leva-se em conta que os agentes podem executar as tarefas em mais de um nível de eficiência. Outra é o *Problema Generalizado de Atribuição Estocástico* (Albareda-Sambola *et al*., 2002). Neste caso, a natureza estocástica do problema pode ser devida à falta de informação a priori sobre a quantidade de recursos necessários para os agentes realizarem as tarefas (o que acontece, por exemplo, com o tempo necessário para programadores executarem tarefas de desenvolvimento de *software*), ou à incerteza sobre a presença ou a ausência de tarefas individuais, ou seja, quando existe um conjunto potencial de tarefas, mas somente algumas delas serão realmente realizadas (o que acontece, por exemplo, em serviços de emergência).

Uma revisão geral das técnicas de solução propostas para o PGA pode ser vista em (Cattrysse e Van Wassenhove, 1992). Alfandari *et al*. (2001) comentam sobre alguns métodos aproximados e exatos já propostos para o PGA. Com relação aos métodos aproximados, em geral, as abordagens propõem heurísticas de duas fases, em que numa primeira fase, as atribuições são calculadas de modo a satisfazer um determinado critério (por exemplo, minimizar uma função de penalização, como em Martello e Toth, 1981) e, numa segunda fase, as atribuições são melhoradas. Amini e Racer (1994) também propõem uma abordagem heurística em duas fases para o problema e fazem uma cuidadosa comparação de vários métodos. Metaheurísticas também já foram propostas para o PGA, dentre as quais podem ser citados os algoritmos genéticos (Wilson, 1997; Chu e Beasley, 1997), a busca tabu (Laguna *et al*., 1995) e *simulated annealing* (Osman, 1995).

Os primeiros algoritmos exatos desenvolvidos para o PGA baseavam-se no método *Branch-and-Bound* (B&B), combinados com relaxação lagrangeana e com heurísticas para a obtenção de limites. É o caso, por exemplo, do algoritmo de Ross e Soland

(1975), com cálculo de limites superiores a partir da relaxação de Programação Linear (PL) do problema de atribuição (ou seja, o PGA sem as restrições de capacidade) e com melhorias locais para atribuições de tarefas. Outro exemplo é o método de busca em profundidade de Fisher *et al*. (1986), que utiliza relaxação lagrangeana das restrições de atribuição. Métodos baseados em heurísticas duais e PL para a obtenção de limitantes superiores e inferiores para o problema primal também já foram propostos. É o caso, por exemplo, do método exato de Guignard e Rosenwein (1989), que se baseia em limites obtidos a partir do dual lagrangeano. Abordagens mais recentes para o PAG propõem a combinação do algoritmo B&B com métodos de Geração de Colunas (GC), o que passou a ser conhecido como algoritmo *Branch-and-Price* (B&P). Barnhart *et al*. (1998) propõem o uso de algoritmos B&P para solucionar problemas inteiros de grande escala e fazem uma revisão de alguns trabalhos similares já propostos na literatura. A relaxação linear e a relaxação lagrangeana têm sido utilizadas para obter limitantes na busca em árvore de algoritmos B&P. Savelsbergh (1997) propõe um algoritmo B&P para resolver uma reformulação do PGA como um Problema de Particionamento de Conjuntos. Pigatti *et al*. (2004) propõem a utilização de cortes combinados com um algoritmo B&P para obter soluções eficientes para o PGA. Trabalhos recentes, publicados por Yagiura *et al*. (2004a e 2004b) propõem outras linhas de investigação para o PGA.

Narciso (1998) aplicou a relaxação lagrangeana/*surrogate* combinada com otimização por subgradientes ao PGA. Os resultados mostram que, para problemas de grande porte, o emprego da relaxação lagrangeana/*surrogate* leva a algoritmos que despendem tempo computacional menor do que quando é empregada a relaxação lagrangeana. Outra abordagem, apresentada por Nauss (2003), propõe a utilização de cortes, relaxação lagrangeana e otimização por subgradientes. Com base nestes resultados, pretende-se, neste trabalho, comparar o desempenho da relaxação lagrangeana com o da relaxação lagrangeana/*surrogate* para a solução do PGA.

O outro problema de atribuição a ser abordado neste trabalho é o PAACC. Para este problema pretende-se propor métodos que encontrem as distribuições das antenas celulares entre os comutadores com custo mínimo. Este problema difere do PGA em

conseqüência da necessidade de se considerar novos custos que surgem devidos às transferências de chamadas entre diferentes regiões, o que torna o problema mais difícil.

O primeiro trabalho encontrado na literatura sobre o PAACC é de Merchant e Sengupta (1995). Nesse trabalho, os autores formulam o PAACC como um problema de Programação Inteira e o resolvem através de heurísticas gulosas de atribuição das células aos comutadores. A mesma abordagem é proposta por Saha *et al*. (2000). Abordagens que utilizam metaheurísticas também já foram propostas para o PAACC. Podem ser citados os trabalhos de Pierre e Houéto (2002), que usaram a busca tabu, de Quintero e Pierre (2002), que empregaram *simulated annealing* (SA) e algoritmos genéticos, e de Menon e Gupta (2004), que combinaram SA com um método de geração de colunas.

Este capítulo apresentou o problema de atribuição em suas diversas formas e estabeleceu os objetivos gerais que se pretende com este trabalho. Em resumo, pretende-se:

- Estudar e propor métodos de solução para dois problemas difíceis de atribuição, a saber: o Problema Generalizado de Atribuição, um problema clássico e para o qual já existem diversas abordagens propostas, e o Problema de Atribuição de Antenas Celulares a Comutadores, um problema mais recente e para o qual ainda existem poucas propostas de solução;

- Comparar o desempenho da aplicação da relaxação lagrangeana/*surrogate* em métodos de solução para o PGA e para o PAACC em relação à utilização da relaxação lagrangeana tradicional.

Este trabalho está organizado da forma descrita a seguir. O Capítulo 2 apresenta uma revisão de literatura para o PGA e para o PAACC, descendo mais a detalhes sobre as técnicas de solução já propostas para estes problemas, em especial, as técnicas que se aproximam das propostas neste trabalho.

O Capítulo 3 apresenta as técnicas envolvidas nos métodos de resolução propostos neste trabalho. Num primeiro momento apresenta-se a técnica de GC, as relaxações lagrangeana e lagrangeana/*surrogate* e as regras de ramificação envolvidas no algoritmo B&P para o caso de problemas inteiros em geral. Em seguida, apresentam-se os métodos propostos para o PGA e para o PAACC, que compreendem a técnica de GC combinada com a relaxação lagrangeana/*surrogate* e, especificamente para o PGA, um método B&P.

O Capítulo 4 descreve os algoritmos implementados neste trabalho, tanto para o PGA como para o PAACC, e apresenta os resultados computacionais obtidos comparando o uso da relaxação lagrangeana/*surrogate* com o uso da relaxação lagrangeana tradicional.

Por fim, o Capítulo 5 apresenta uma síntese do trabalho, analisa e discute os resultados obtidos, comenta sobre algumas dificuldades encontradas e sugere temas para pesquisas futuras.

CAPÍTULO 2

REVISÃO DE MÉTODOS PROPOSTOS PARA O PGA E PAACC

Este capítulo discute algumas das técnicas já propostas para o Problema Generalizado de Atribuição e para o Problema de Atribuição de Antenas Celulares a Comutadores. Para o PGA, como se trata de um problema muito estudado e para o qual existe uma grande diversidade de métodos propostos, discutem-se, em especial, as técnicas que mais se aproximam das propostas neste trabalho. Discute-se em primeiro lugar a técnica de Geração de Colunas aplicada ao PGA e, em seguida, os métodos B&P propostos por Savelsbergh (1997) e por Pigatti (2003). Em seguida, ainda para o PGA, apresenta-se o método proposto por Narciso e Lorena (1999), que faz uso da relaxação lagrangeana/*surrogate.*

Para o PAACC, por se tratar de um problema relativamente recente, apresentam-se as principais propostas de resolução encontradas na literatura. Discute-se a abordagem heurística proposta por Merchant e Sengupta (1995) para o PAACC em sua forma original e com variação das demandas de chamadas. Em seguida, são apresentados o algoritmo de Busca Tabu proposto por Pierre e Houéto (2002), as abordagens de Algoritmo Genético e de *Simulated Annealing* apresentadas por Quintero e Pierre (2003) e uma abordagem mista proposta por Menon e Gupta (2004), que incorpora técnicas de Geração de Colunas em um algoritmo *Simulated Annealing*.

2.1 O Método de Geração de Colunas para o PGA

Sejam M = {1, ..., m} e N = {1, ..., n}, os conjuntos de índices de máquinas e tarefas, respectivamente. Com isto, a formulação do PGA estabelecida em (1.10)-(1.13) pode ser reescrita como:

$$\min \quad \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \tag{2.1}$$

sujeito a:

$$\sum_{j \in N} r_{ij} x_{ij} \leq b_i \quad \forall\, i \in M \tag{2.2}$$

$$\sum_{i \in M} x_{ij} = 1, \quad \forall\, j \in N \tag{2.3}$$

$$x_{ij} \in \{0, 1\}, \quad \forall\, i \in M,\, j \in N \tag{2.4}$$

O método de GC para o PGA baseia-se na reformulação deste problema como um Problema de Cobertura de Conjuntos (PCC). Para isto, seja $K_i = \{x^i_1, ..., x^i_{k_i}\}$ o conjunto de padrões de atribuição viáveis para a máquina i, onde $x^i_k = \{x^i_{1k}, ..., x^i_{nk}\}$ é uma solução viável para as restrições (2.2) e (2.4). Seja $y^i_k, i \in M \text{ e } k \in K_i$, uma variável binária caracterizando se a atribuição x^i_k está associada realmente à máquina i, ou seja:

$$y^i_k = \begin{cases} 1, \text{ se a atribuição } x^i_k \text{ está associada à máquina } i \\ 0, \text{ caso contrário} \end{cases}$$

Então, o PGA pode ser reformulado como o seguinte PCC:

$$\min \quad \sum_{i=1}^{m} \sum_{k=1}^{k_i} \left(\sum_{j=1}^{n} c_{ij} x^i_{jk} \right) y^i_k \tag{2.5}$$

sujeito a:

$$\sum_{i=1}^{m} \sum_{k=1}^{k_i} x^i_{jk} y^i_k = 1, \quad j \in N \tag{2.6}$$

$$\sum_{k=1}^{k_i} y^i_k \leq 1, \quad i \in M \tag{2.7}$$

$$y^i_k \in \{0,1\}, \quad i \in M \tag{2.8}$$

Esta formulação foi usada por Cattrysse *et al*. (1994) para desenvolver um algoritmo heurístico para o PGA. O problema a ser resolvido pelo método de GC é a versão de PL deste problema de cobertura de conjuntos, ou seja:

$$\min \quad \sum_{i=1}^{m}\sum_{k=1}^{k_i}\left(\sum_{j=1}^{n} c_{ij} x_{jk}^{i}\right) y_k^i \tag{2.9}$$

sujeito a:

$$\sum_{i=1}^{m}\sum_{k=1}^{k_i} x_{jk}^{i} y_k^i = 1, \quad j \in N \tag{2.10}$$

$$\sum_{k=1}^{k_i} y_k^i \leq 1, \quad i \in M \tag{2.11}$$

$$y_k^i \in [0,1], \quad i \in M \tag{2.12}$$

No contexto do método de GC, o problema (2.9)-(2.12) é conhecido como Problema Mestre Restrito (PMR) (Barnhart *et al.*, 1998). Na abordagem tradicional, após definir um conjunto inicial de colunas, o PMR é resolvido e os custos duais finais π_j ($j \in N$), referentes às restrições (2.10), e μ_i ($i \in M$), referentes às restrições (2.11), são usados para gerar novas colunas. Para isto, as atribuições viáveis x_k^i para cada máquina i são obtidas resolvendo-se os seguintes m Problemas da Mochila:

$$\min \quad v(M_i) = \sum_{j=1}^{n} (c_{ij} - \pi_j) x_{ij} \tag{2.13}$$

sujeito a:

$$\sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i \tag{2.14}$$

$$x_{ij} \in \{0, 1\}, \quad j \in N \tag{2.15}$$

Cada solução viável x_k^i corresponde a uma coluna na formulação de cobertura de conjuntos. Essa coluna consiste de n valores iguais a 1 ou 0 representando se a tarefa j está ou não atribuída à máquina i (conforme solução do problema da mochila M_i) e de um vetor unitário e_i, onde i corresponde ao número da máquina, como ilustra a FIGURA 2.1 a seguir.

$$x_k^i = \begin{bmatrix} 1 & \text{tarefa 1} \\ 0 & \text{tarefa 2} \\ \vdots & \\ 1 & \text{tarefa n} \\ 0 & \text{máquina 1} \\ \vdots & \\ 1 & \text{máquina i} \\ \vdots & \\ 0 & \text{máquina m} \end{bmatrix}$$

FIGURA 2.1 – Coluna para o Problema de Cobertura de Conjuntos

Todas as colunas correspondentes a $v(M_i) - \mu_i < 0$ são candidatas a serem selecionadas e podem ser incorporadas ao PMR.

Assim, o método de GC para o PGA pode ser estabelecida pelo seguinte algoritmo:

1) Estabelecer um conjunto inicial de colunas para o PMR;

2) Resolver o problema PMR, obtendo os custos duais π_j ($j \in N$) e μ_i ($i \in M$);

3) Resolver os problemas da mochila M_i, obtendo as potenciais colunas a serem acrescentadas ao PMR;

4) Acrescentar ao PMR colunas correspondentes a $v(M_i) - \mu_i < 0$ ($i \in M$);

5) Parar, se o passo (4) não acrescentar novas colunas ao PMR;

6) Executar os testes de remoção de colunas improdutivas e retornar ao passo (2).

Savelsbergh (1997) sugere, para o passo (4) deste algoritmo, considerar a primeira coluna com custo reduzido negativo encontrada ou, para evitar soluções tendenciosas, escolher uma das colunas candidatas aleatoriamente. Outra alternativa, também proposta, é considerar todas as colunas com custos reduzidos negativos (estratégia conhecida como *multi-pricing* na literatura). Isto, no entanto, pode aumentar demasiadamente o tamanho do PMR.

2.2 O Método B&P Aplicado ao PGA

A menos que todas as variáveis y_k^i sejam inteiras, a solução obtida pelo método de Geração de Colunas não é factível para o PGA. Assim, faz-se necessário aplicar um algoritmo B&B de modo a obter soluções exatas para o problema. Na construção da árvore B&B, se y_k^i é fracionária então uma opção é fazer $y_k^i = 0$ em um ramo e $y_k^i = 1$ em outro. Se $y_k^i = 0$, independente do valor do elemento x_{jk}^i que compõe a atribuição x_k^i, a tarefa j deverá ser atribuída a uma máquina diferente de i. No caso em que $y_k^i = 1$, a tarefa j será atribuída a i somente se $x_{jk}^i = 1$; caso contrário, a tarefa j também deverá ser atribuída a outra máquina. Em resumo, em uma ramificação a tarefa j não será de maneira alguma atribuída à máquina i e na outra ramificação pode ser ou não atribuída à máquina i. Uma outra opção é fazer $x_{ij} = 0$ em um ramo e $x_{ij} = 1$ em outro, ou seja, considerar em um ramo apenas as atribuições em que a tarefa j está atribuída à máquina i e, no outro, as demais atribuições possíveis da tarefa j.

Savelsbergh (1997) explorou duas estratégias de ramificação para o método B&P. Na primeira, considerou a variável fracionária x_{ij} mais próxima de 0.5, fixando $x_{ij} = 1$ em um ramo e $x_{ij} = 0$ em outro ramo. No caso de empate, escolhia-se a variável com menor custo c_{ij}. A segunda estratégia consiste em verificar a somatória $\sum_{i=1}^{m} x_{ij}\ (j = 1, ..., n)$ com

o maior número de variáveis fracionárias, fazendo $\sum_{i=1}^{i^*} x_{ij} = 0$ em um ramo e $\sum_{i=i^*+1}^{m} x_{ij} = 0$ no outro ramo, onde o índice i* é escolhido tão próximo quanto possível de m/2 e de forma que deve existir pelo menos uma variável fracionária em ambos os ramos. No caso de empate, seleciona-se a restrição para a qual o valor de $\left| \sum_{i=1}^{i^*} x_{ij} - 0.5 \right| + \left| \sum_{i=i^*+1}^{m} x_{ij} - 0.5 \right|$ é o menor possível.

Para a busca, Savelsbergh (1997) também usou duas estratégias: busca em profundidade e busca do melhor limite. A busca em profundidade é aplicada para obter boas soluções mais rapidamente, pois com uma boa solução é possível podar um número significativo de nós e assim reduzir o tamanho da árvore. A estratégia de busca do melhor limite explora primeiro o nó com o limite mais promissor (ainda não explorado).

Para a Geração de Colunas, Pigatti (2003) utilizou o algoritmo de estabilização proposto por du Merle *et al.* (1999), evitando assim que a convergência se torne lenta quando os valores das variáveis duais associadas oscilam e o subproblema de geração de colunas passa a obter, a cada iteração, colunas com características muito diferentes. A seguir é descrita a proposta de Pigatti (2003).

Seja um programa linear P viável e limitado:

$$\min \sum_{j=1}^{n} c_j x_j \tag{2.16}$$

sujeito a:

$$\sum_{j=1}^{n} \alpha_{ij} x_j = b_i, \quad i = 1, ..., m \tag{2.17}$$

$$x_j \geq 0, \; j = 1,..., n \tag{2.18}$$

e o seu dual D:

$$\max \sum_{i=1}^{m} b_i \pi_i \tag{2.19}$$

sujeito a:

$$\sum \alpha_{ij} x_j \leq c_j \quad j = 1,..., n \tag{2.20}$$

Pigatti (2003) reduziu a degenerescência, aplicando uma perturbação em P através da adição de variáveis de excesso y_1 e de variáveis de folga y_2 limitadas por valores ε_1 e ε_2 pequenos, respectivamente. Desta forma evita-se que as variáveis duais oscilem através do problema perturbado (P_P), dado por:

$$\min \sum_{j=1}^{n} c_j x_j \tag{2.21}$$

sujeito a:

$$\sum_{j=1}^{n} \alpha_{ij} x_j - y_{1i} - y_{2i} = b_i, \; i = 1, ..., m \tag{2.22}$$

$$x_j \geq 0, \; j = 1,..., n \tag{2.23}$$

$$0 \leq y_{1i} \leq \varepsilon_1, 0 \leq y_{2i} \leq \varepsilon_2, \; i = 1, ..., m \tag{2.24}$$

A solução de P_p é menor ou igual do que a solução do problema original (P) e, portanto, é um limite inferior válido para o problema inteiro. Fazendo-se os valores de ε_1 e ε_2 suficientemente pequenos, essa perda se torna pouco significativa.

O problema (P_p) tem o dual (D_P):

$$\max \sum_{i=1}^{m} b_i \pi_i - \sum_{i=1}^{m} \omega_{1i} \varepsilon_1 + \sum_{i=1}^{m} \omega_{2i} \varepsilon_2 \tag{2.25}$$

sujeito a:

$$\sum \alpha_{ij} x_j \leq c_j \quad j = 1,..., n \tag{2.26}$$

$$\omega_{2i} \geq \pi_i \, , \, i = 1, ..., m \tag{2.27}$$

$$\omega_{1i} \geq -\pi_i \, , \, i = 1, ..., m \tag{2.28}$$

O espaço dual de (D_P) é dado na FIGURA 2.2 e observa-se que os valores das variáveis duais estão limitados a um intervalo [d_1, d_2].

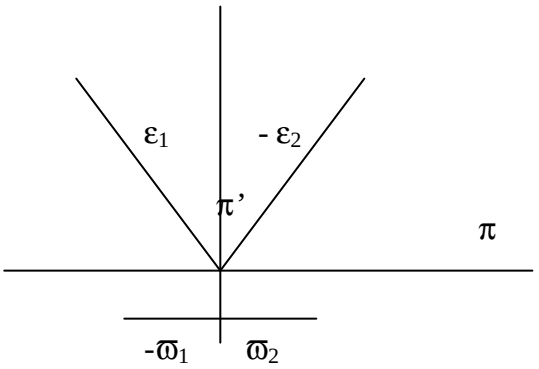

FIGURA 2.2 – Espaço dual do problema (Dp)
(FONTE: Pigatti (2003), p. 36)

Uma outra maneira de tentar acelerar a convergência da geração de colunas, evitando que as variáveis duais oscilem excessivamente ao longo das iterações, é feita através de restrições que limitam o valor das variáveis duais a um intervalo de valores [d_1; d_2], ou seja, considere o problema D_r, dado por:

$$\max \sum_{i=1}^{m} b_i p_i \tag{2.29}$$

sujeito a:

$$\sum_{i=1}^{m} \alpha_{ij} \pi_i \leq c_j \;\; j = 1, \ldots, n \tag{2.30}$$

$$d_{1i} \leq \pi_i \leq d_{2i} \;\; i = 1, \ldots, m \tag{2.31}$$

Seu dual corresponde ao seguinte problema primal P_d:

$$\min \sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} d_{1i} y_{1i} + \sum_{i=1}^{m} d_{2i} y_{2i} \tag{2.32}$$

sujeito a:

$$\sum_{j=1}^{n} \alpha_{ij} x_j - y_{1i} + y_{2i} = b_i \, , \; i = 1, \ldots, m \tag{2.33}$$

$$x_j \geq 0 \, , \; j = 1, \ldots, n \tag{2.34}$$

$$y_{1i} \, , y_{2i} \geq 0 \, , \; i = 1, \ldots, m \tag{2.35}$$

Pigatti (2003) faz, a cada iteração, ajustes nos valores de π, para encontrar uma solução dual que esteja contida na faixa dada por d_1 e d_2 e, conseqüentemente, procurando com que o valor da solução desse novo problema se aproxime da solução do problema original. Este procedimento consiste no algoritmo *Boxstep* ilustrado na FIGURA 2.3.

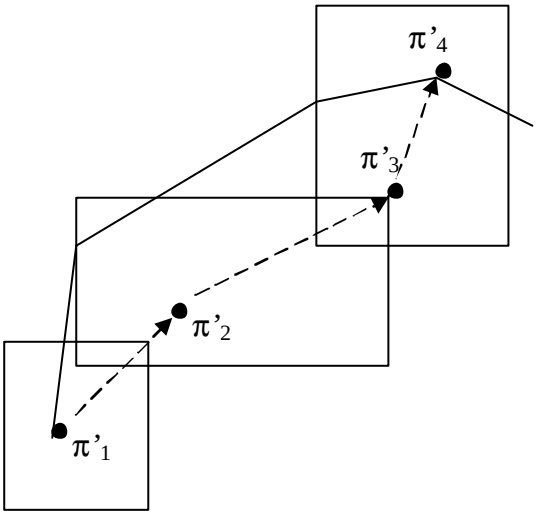

FIGURA 2.3 – Algoritmo *Boxstep*

(FONTE: Pigatti (2003), p. 37)

A formulação do método de Geração de Colunas consiste na combinação de P_p e P_d , ou seja, no problema P_e dado por:

$$\min \sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} d_{1i} y_{1i} + \sum_{i=1}^{m} d_{2i} y_{2i} \tag{2.36}$$

sujeito a:

$$\sum_{j=1}^{n} \alpha_{ij} x_j - y_{1i} + y_{2i} = b_i \quad i = 1, \ldots, m \tag{2.37}$$

$$0 \leq y_{1i} \leq \varepsilon_1, \; i = 1, ..., m \tag{2.38}$$

$$0 \leq y_{2i} \leq \varepsilon_2, \; i = 1, ..., m \tag{2.39}$$

$$x_j \geq 0, \; j = 1, \ldots, n \tag{2.40}$$

Cujo dual D_e associado a P_e é como segue:

$$\max \sum_{i=1}^{m} b_i \pi_i - \sum_{i=1}^{m} \varpi_{1i} \varepsilon_1 - \sum_{i=1}^{m} \varpi_{2i} \varepsilon_2 \tag{2.41}$$

sujeito a:

$$\sum_{i=1}^{m} \alpha_{ij} \pi_i \leq c_j, \; j = 1, \ldots, n \tag{2.42}$$

$$d_{1i} - \varpi_{1i} \leq \pi_i \leq d_{2i} + \varpi_{2i}, \; i = 1, \ldots, m \tag{2.43}$$

$$\omega_{1i} \geq 0 \text{ e } \omega_{2i} \geq 0, \; i = 1, ..., m \tag{2.44}$$

Para inicializar π', Pigatti (2003) usou o valor das variáveis duais da formulação clássica do PGA relaxado e fez $d_1 = d_2 = \pi$' com os valores 0.1, 0.01, 0.001 e 0 para ε. Em seus testes, Pigatti (2003) observa que quando $\varepsilon = 0.1$, qualquer mudança nos valores das variáveis duais é muito penalizada e que para $\varepsilon = 0.01$, a penalização pela mudança no valor da variável dual diminui, mas o processo converge rapidamente porque foram geradas boas colunas para o caso em que $\varepsilon = 0.1$ e estas colunas são mantidas no problema. Para $\varepsilon = 0.001$, a penalização por mudança no valor da variável

dual diminui ainda mais. Finalmente, para ε igual a 0, ou seja, quando não há penalização pela troca de valor da variável dual, a convergência é rápida devido à qualidade das colunas que foram geradas nas três execuções anteriores. O procedimento de estabilização é dado pelo seguinte algoritmo:

1) Faça: π = duais da relaxação e $\varepsilon = 0.1$;
2) valRetorno = geraColunas();
3) Se valRetorno $\geq$ *cuttoff* $- 1 +$ eps, então Pare.
4) i = 0;
5) Faça π = novos duais calculados pelo procedimento geraColunas;
6) Se i = 2 então $\varepsilon = 0$, senão $\varepsilon = (0.1)^{i+2}$;
7) valRetorno = geraColunas();
8) Se valRetorno $\geq$ *cuttoff* $- 1 +$ eps, então Pare;
9) Se i $\neq$ 2, então faça i = i + 1 e retorne ao passo 5.

onde:

- π são os valores das variáveis duais correspondentes à penalidade zero;
- ε é a penalização pelas variáveis duais a partir de π;
- geraColunas é o procedimento de geração de colunas;
- *cutoff* é um valor limite do procedimento de *Branch-and-Bound*. Se o valor retornado estiver acima do *cutoff* o procedimento de geração de colunas deve parar.

Em seu trabalho, Pigatti (2003) faz a escolha da variável a ser fixada na ramificação levando em consideração o quanto a variável fracionária está próxima de 0.5 e o seu custo. Se dist é a distância entre 0.5 e a variável e c o custo da variável, a escolha é feita

pela variável com o menor valor para (1 – dist) * 100 + c, fixando-a em 0 e, em seguida, em 1. O algoritmo de ramificação e geração de novas colunas, ou seja, o algoritmo de *Branch-and-Price*, é dado por:

1) valSolucao = gerarColunasEstabilizadas(valMelhorSolucao);
2) Se valSolucao ≥ valMelhorSolucao – 1 + ε, então Pare;
3) Execute determinarPróximoFixado(i, j);
4) Se i = 0 e j = 0 (solução inteira encontrada), então:

    Se not (usandoColunas()), então:

        Se valSolucao < valMelhorSolucao, então:

            valMelhorSolucao = valSolucao;

    Pare.

    Fim_Se;
5) Execute retirarColunasViolamZero(i, j) e fixarZero(i, j);
6) ChamarAlgoritmoRecursivamente();
7) Execute recolocarColunasViolamZero(i, j) e desFixarZero(i, j);
8) Execute retirarColunasViolamUm(i, j) e fixarUm(i, j);
9) chamarAlgoritmoRecursivamente();
10) Execute recolocarColunasViolamUm(i, j) e desFixarUm(i, j).

onde:

– gerarColunasEstabilizadas é o procedimento de geração de colunas estabilizadas descrito anteriormente;

- valMelhorSolucao é o valor da melhor solução inteira encontrada;
- determinaProximoFixado é uma função que determina a próxima variável x_{ij} a ser fixada. Se a função retornar i = 0 e j = 0 significa que a solução corrente é inteira;
- usandoColunas é uma função que retorna verdadeiro se alguma das colunas artificiais inseridas pelo problema mestre inicial está sendo usada na solução inteira corrente. Caso esteja usando, a solução encontrada é inviável;
- retirarColunasViolamZero é uma função que retira do modelo todas as colunas que violam a fixação de x_{ij} em 0;
- fixarZero é uma função que fixa x_{ij} em 0;
- chamarAlgoritmoRecursivamente() representa a chamada do procedimento de B&P recursivamente;
- recolocarColunasViolamZero é uma função que recoloca no modelo as colunas que foram retiradas pela função retirarColunasViolamZero;
- desFixarZero é uma função que retira a fixação de x_{ij} em 0;
- retirarColunasViolamUm é uma função que retira do modelo todas as colunas que violam a fixação de x_{ij} em 1;
- fixarUm é uma função que fixa x_{ij} em 1;
- recolocarColunasViolamUm é uma função que recoloca no modelo as colunas que foram retiradas pela função retirarColunasViolamUm;
- desFixarUm é uma função que retira a fixação de x_{ij} em 1.

2.3 Uma Proposta Empregando a Relaxação Lagrangeana/*Surrogate*

Narciso (1998) propõe utilizar a relaxação lagrangeana/*surrogate* para a solução do PGA. O algoritmo utilizado é uma versão generalizada do algoritmo de subgradientes.

Considere que o dual lagrangeano/*surrogate* do problema (2.1)-(2.4) é dado por:

$$v(LS_t^{\,x}) = \max_{t\geq 0} \min_{x\in S} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + t\sum_{i=1}^{m}\lambda_i\left(b_i - \sum_{j=1}^{n} r_{ij}x_{ij}\right) \qquad (2.45)$$

O algoritmo para solução do PGA é então estabelecido como:

1) Para j = 1, ..., m, faça $\lambda_i = \left(\dfrac{\sum_{j=1}^{n} r_{ij} - b_i}{\sum_{j=1}^{n} r_{ij}} \right)$;

2) Resolva a relaxação lagrangeana/*surrogate* dada pelo problema (2.45) e obtenha x^{λ};

3) Obtenha uma solução factível x^f a partir de x^{λ} usando heurísticas construtivas e calcule seu custo como: $v_f = \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij}^{f}$. Faça: $l_{inf} = \max[\, l_{inf}, v_f\,]$ e $l_{sup} = \min[\, l_{sup}, v(R_{\lambda})]$, onde $v(R_\lambda) = v(LS_t^{\,x\lambda})$;

4) Atualize a direção do subgradiente g^{λ}, ou seja, para i = 1, ..., m, faça: $g_i^{\lambda} = b_i - \sum_{j=1}^{n} r_{ij}x_{ij}^{\lambda}$. Atualize também o tamanho do passo t.

5) Para i = 1, ..., m, faça $\lambda_i = \max\{0, \lambda_i + t\, g_i^{\lambda}\}$;

6) Retorne ao passo 2.

Em seu trabalho, Narciso (1998) conclui que a relaxação lagrangeana/*surrogate* obtém limites tão bons quanto os fornecidos pela relaxação lagrangeana, mas em tempo

computacional menor, principalmente quando as instâncias têm grandes dimensões. Esta conclusão aparece também em Narciso e Lorena (1999).

2.4 O Problema de Atribuição de Antenas Celulares a Comutadores

Nas três últimas décadas houve um aumento significativo dos sistemas de comunicação móvel. Geralmente, uma área geográfica atendida por serviços de comunicação móvel é dividida em unidades geográficas menores denominadas células. Para efeito de estudo, as células são, em geral, representadas por hexágonos, conforme mostra a FIGURA 2.4.

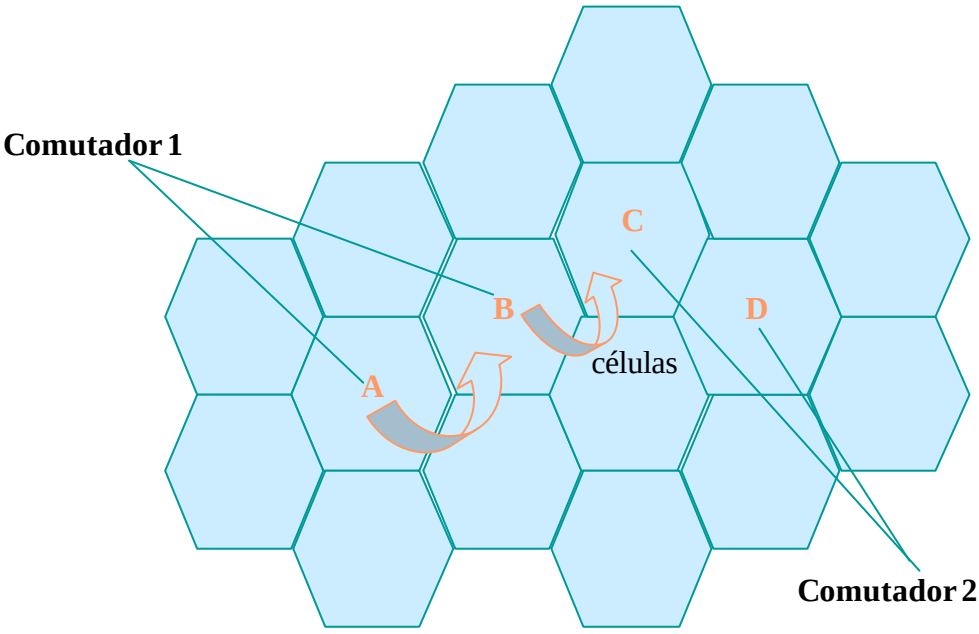

FIGURA 2.4 – Representação de Células em uma Rede de Comunicação

(FONTE: Merchant e Sengupta (1995), p. 521)

Cada célula possui uma antena de transmissão e recepção, também conhecida como estação rádio base, para fazer a comunicação entre o equipamento celular móvel e um comutador, que tem a função de encaminhar o tráfego de chamadas entre antenas.

Durante uma chamada, conforme aumenta a distância entre a unidade móvel do assinante (celular) e a estação rádio base, o sinal torna-se fraco e aumentam os ruídos provenientes de interferências da vizinhança. Para evitar isto, o sistema faz uma transferência da chamada de uma antena para outra adjacente que esteja mais próxima do usuário. Esta transferência (denominada *handoff*) deve ser realizada para que o assinante não perca a qualidade do sinal enquanto estiver realizando uma chamada. Quando o assinante se move entre antenas atendidas por um mesmo comutador, a transferência é conhecida como *handoff* simples. Quando o assinante se move entre antenas atendidas por comutadores diferentes tem-se então o denominado *rooming* ou *handoff* complexo. Isto pode ser ilustrado pela FIGURA 3.1: quando durante uma chamada um assinante se move da célula A para célula B, que são atendidas pelo Comutador 1, tem-se o *handoff* simples e quando o assinante sai da célula B para a célula C, atendidas pelo Comutador 1 e Comutador 2, respectivamente, tem-se o *handoff* complexo, que envolve um custo maior. Em geral, os custos de *handoff* simples são desconsiderados.

O PAACC consiste em determinar qual a maneira ótima de se atribuir n células a m comutadores com posições fixas em uma dada região, de forma a minimizar os custos de cabeamento entre antenas e comutadores e os custos de *handoff* complexos. Caso sejam desconsiderados os custos de transferência de chamadas entre diferentes regiões, o PAACC recai na formulação do PGA.

Sejam M = { 1, ..., m } e N = { 1, ..., n }. Para formular o PAACC são assumidas como conhecidas as seguintes informações:

- os custos c_{ik} de cabeamento entre as células i e os comutadores k, com $i \in N$ e $k \in M$;
- os custos h_{ij} de *handoff* por unidade de tempo entre as células i e j, com $i, j \in N$;
- o volume de chamadas λ_i de cada célula i, com $i \in N$;

- a capacidade de atendimento de chamadas M_k de cada comutador k, com $k \in M$.

Além disto, definem-se as seguintes variáveis de decisão:

- $x_{ik} = \begin{cases} 1, \text{se a célula i é atribuída ao switch k} \\ 0, \text{caso contrário.} \end{cases}$, com $i \in N$ e $k \in M$, que representam a atribuição ou não da célula i ao comutador k;

- $z_{ijk} = \begin{cases} 1, \text{se as células i e j são atribuídas ao switch k} \\ 0, \text{caso contrário.} \end{cases}$, com $i \in N$ e $k \in M$, que exprimem o fato de duas antenas celulares i e j estarem ou não atribuídas ao mesmo comutador k; e

- $y_{ij} = \begin{cases} 1, \text{se as células i e j são atribuídas a um mesmo switch} \\ 0, \text{caso contrário.} \end{cases}$, com $i, j \in N$, que representam a possibilidade de duas células i e j estarem atribuídas ou não a um mesmo comutador.

Com isto, o PAACC pode ser formulado como:

$$\min \quad \sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik}x_{ik} + \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(1 - y_{ij}) \tag{2.46}$$

sujeito a:

$$\sum_{k=1}^{m} x_{ik} = 1 \quad \forall i \in N \tag{2.47}$$

$$\sum_{i=1}^{n} \lambda_i x_{ik} \leq M_k \quad \forall k \in M \tag{2.48}$$

$$z_{ijk} = x_{ik}x_{jk} \quad \forall i, j \in N \text{ e } k \in M \tag{2.49}$$

$$y_{ij} = \sum_{k=1}^{m} z_{ijk} \quad \forall\, i, j \in N \tag{2.50}$$

$$x_{ik}, y_{ij} \in \{0,1\} \quad \forall\, i, j \in N, k \in M \tag{2.51}$$

A função-objetivo (2.46) busca minimizar os custos de cabeamento e de *handoff*. As restrições de atribuição (2.47) evitam que uma célula seja atribuída a mais de um comutador. As restrições de capacidade (2.48) impossibilitam que as capacidades de atendimento M_k dos comutadores sejam violadas. As restrições (2.49) implicam que a variável z_{ijk} assume o valor 1 se e somente se, as células i e j estiverem atribuídas ao mesmo comutador k. As restrições (2.50) implicam que y_{ij} assume o valor 1 se e somente se, as antenas celulares i e j estiverem atribuídas a um único comutador. Finalmente, as restrições (2.51) correspondem às condições de integralidade das variáveis de decisão.

Dessa maneira, o PAACC é um problema de programação não-linear inteira porque possui variáveis inteiras e as restrições (2.49) são não lineares. No entanto, como observado em Merchant e Sengupta (1995), estas restrições podem ser substituídas por:

$$z_{ijk} \leq x_{ik} \tag{2.52}$$

$$z_{ijk} \leq x_{jk} \tag{2.53}$$

$$z_{ijk} \geq x_{ik} + x_{jk} - 1 \tag{2.54}$$

$$z_{ijk} \geq 0 \tag{2.55}$$

Diz-se que o PAACC tem *homing* singular, quando a demanda de chamadas permanece inalterada. Caso a demanda de chamadas varie no mesmo período, o problema é considerado como tendo *homing* dual. Neste caso, uma célula pode ser conectada a diferentes comutadores, dependendo da necessidade do período. A permissão de atribuição de uma célula a mais de um comutador deve ser considerada somente se reduzir os custos de *handoff*, uma vez que os custos de cabeamento serão aumentados.

Nesta versão do problema existem dois instantes diferentes em um período, com volumes de chamadas e custos de *handoff* diferentes. Assim, adicionam-se os volumes de chamada $\overline{\lambda}_i$ e os custos de *handoff* $\overline{h}_{ij}$ relativos às segundas demandas do período. Os valores de M_k e c_{ik} permanecem inalterados.

Merchant e Sengupta (1995) consideram a busca por dois padrões de atribuição, ambos correspondendo a *homings* singulares. Se a célula i está atribuída ao comutador k em ambos os padrões, seu custo de cabeamento estará duplicado. Seja para o padrão 1, com as variáveis x_{ik}, y_{ij} e z_{ijk} satisfazendo as restrições do problema. Para o padrão 2, definem-se as variáveis correspondentes $\overline{x}_{ik}$, $\overline{y}_{ij}$ e $\overline{z}_{ijk}$ satisfazendo as restrições do PAACC trocando-se λ_i por $\overline{\lambda}_i$. Para que o custo de cabeamento não seja considerado duas vezes no caso da célula ser atribuída ao mesmo comutador nos dois padrões, define-se a seguinte variável:

$$w_{ik} = x_{ik} \vee \overline{x}_{ik} \quad \forall i \in N, k \in M \tag{2.56}$$

onde o símbolo "$\vee$" significa a operação "ou". Dessa forma, a função-objetivo do problema pode ser escrita como:

$$\sum_{i=1}^{n}\sum_{k=1}^{m} c_{ik} w_{ik} + \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(1-y_{ij}) + \sum_{i=1}^{n}\sum_{j=1}^{n} \overline{h}_{ij}(1-\overline{y}_{ij}) \tag{2.57}$$

Com esta nova variável, o conjunto de restrições de linearização (2.52)-(2.55) pode ser substituído por:

$$w_{ik} \geq x_{ik} \tag{2.58}$$

$$w_{ik} \geq \overline{x}_{ik} \tag{2.59}$$

$$w_{ik} \leq 1 \tag{2.60}$$

$$w_{ik} \leq x_{ik} + \overline{x}_{ik} \tag{2.61}$$

2.5 Uma Abordagem Heurística para o PAACC

Apresenta-se a seguir a proposta de Merchant e Sengupta (1995) para o PAACC. Considera-se que uma atribuição de células é factível se a atribuição satisfaz as restrições de capacidades (2.48) dos comutadores e que movimentos em uma atribuição factível são factíveis se a atribuição resultante também for factível. Para gerar uma atribuição inicial factível, ordenam-se as células em ordem decrescente em relação ao volume de chamadas (λ_i). O algoritmo é dividido em m estágios. No estágio k, a atribuição das k-1 primeiras antenas celulares já foram definidas e não podem ser mudadas. A k^a célula (k = 1, 2, ..., n) é então atribuída de maneira que o comutador escolhido não tenha sua capacidade violada (atribuição factível) e resulte nos menores custos de *handoff* e cabeamento. Em seguida, estende-se esta idéia de modo a que sejam obtidas b soluções factíveis para o problema. Para isto, considere que existem b soluções factíveis no (k-1)^o estágio do problema. Para k = 2, isto é trivial, mas no k^o estágio, m escolhas de atribuição da célula i são possíveis. Dentre todas as possibilidades retiram-se as infactíveis e retêm-se apenas as b possibilidades com menores custos. Repete-se o processo para k = 1, 2, ..., n. A heurística seleciona então a melhor atribuição e usa um mecanismo de refinamento para melhorar o resultado de maneira seqüencial. O mecanismo de refinamento aplica repetidamente movimentos factíveis que forneçam o melhor ganho para a função-objetivo, até que um critério heurístico de otimalidade seja alcançado. Nos testes realizados utilizou-se b = 10.

Esta abordagem é descrita a seguir, apresentando em algoritmos separados, as várias fases do processo. Inicialmente são determinadas as atribuições iniciais, com o seguinte algoritmo:

2.5.1 Algoritmo de Atribuição Inicial

1) Ordene as células em ordem decrescente em relação ao volume de chamadas (λ_i), iniciando com uma atribuição vazia.

2) As células são atribuídas uma de cada vez em estágios. Para k = 1, 2, ..., n, faça:

a) Estenda cada atribuição parcial considerando a adição de todas as atribuições possíveis da k^a célula, descartando todas que violem as restrições de capacidade de atendimento dos comutadores. Se nenhuma atribuição é válida pare, pois o algoritmo falhou. Se restarem menos do que b atribuições parciais, armazene todas; caso contrário, armazene somente as b melhores atribuições, considerando os custos de *handoffs* e cabeamento da k^a célula atribuída somente.

3) Retorne a melhor das atribuições encontradas.

Com a solução inicial obtida, procura-se reduzir o valor da função-objetivo através de movimentos factíveis, como mostrado no algoritmo a seguir.

2.5.2 Algoritmo de Refinamento

Para refinar a solução na atribuição inicial, foi utilizado o seguinte algoritmo, que se repete até que nenhum passo reduza o valor da função-objetivo:

1) Marque todas as células como desligadas.

2) Encontre o melhor movimento factível como segue: escolha uma célula i e um comutador k tal que a atribuição desta célula a este comutador reduza a função-objetivo ao máximo.

3) Atribua a célula i ao comutador k. Marque a célula i como ligada. Anote a atribuição atual.

4) Repita os passos 2 e 3 até que nenhum movimento factível seja encontrado. Em seguida, selecione o padrão de atribuição com menor valor objetivo e finalize.

5) Se o valor da função objetivo não se alterou então pare. Caso contrário retorne ao passo 1.

A heurística para *homing* dual baseia-se na resolução de diferentes *homings* singulares e, a partir delas, constrói-se um segundo conjunto de atribuições que sejam factíveis

para o *homing* dual. Tomam-se duas atribuições, uma atribuição de cada conjunto referente a *homings* singulares. Verifica-se então, célula por célula, se existe alguma atribuída ao mesmo comutador, caso em que se considera o custo de cabeamento apenas uma vez. Faz-se esta verificação repetidamente até que todas atribuições tenham sido consideradas. Apresenta-se a seguir a descrição do algoritmo.

2.5.3 Algoritmo para *Homing* Dual

1) Formule um PAACC com *homing* singular, denotado por Q, considerando o padrão 1 (λ_i, h_{ij}, M_k, c_{ik}) com i, j $\in$ N e k $\in$ M. Formule um segundo PAACC com *homing* singular, denotado por Q', considerando o padrão 2 (λ'_i, h'_{ij}, M_k, c_{ik}) com i, j $\in$ N e k $\in$ M.

2) Resolva ambos os problemas usando um algoritmo de *homing* singular. Seja A a solução do problema Q com o menor valor da função-objetivo.

3) Seja Q_1 o problema de *homing* singular idêntico a Q', considerando a seguinte modificação: se A atribui a célula i ao comutador k, então elimine o custo de cabeamento c_{ik} em Q_1. Resolva Q_1 e chame esta solução de A'.

4) Similarmente, faça Q_2 o problema de *homing* singular idêntico a Q, com a seguinte alteração: se A' atribui a célula i ao comutador k, então elimine o custo de cabeamento c_{ik} em Q_2. Resolva Q_2 e chame esta solução de A.

5) Repita os passos (3) e (4) até que as atribuições dadas não melhorem a função-objetivo. As atribuições A e A' combinadas formam uma solução para o *homing* dual.

2.6 Uma Abordagem de Busca Tabu para o PAACC

A Busca Tabu (BT) foi adaptada aos problemas de otimização combinatória por Glover *et al*. (1993). É uma metaheurística de melhoria da solução que fornece meios de explorar o espaço de soluções em um sentido mais amplo, indo na direção de pontos além dos ótimos locais. Na BT, cria-se uma lista (denominada lista tabu) para

armazenar informações que caracterizam os movimentos realizados em uma vizinhança de uma solução corrente. Esta lista tem a função de evitar que uma solução ou uma seqüência de soluções seja visitada repetidas vezes. Concomitantemente, estabelece-se um mecanismo de avaliação para determinar a aceitação ou não dos movimentos que levem às novas soluções. Desse modo, a busca fica restrita por uma estratégia de proibição, que utiliza e controla a lista tabu, evitando o retorno às soluções já visitadas e, dessa forma, induzindo a exploração de novas regiões. A BT pode até aceitar movimentos locais que não melhorem a solução, mas que possam levar ao ótimo global.

O algoritmo de BT para o PAACC proposto por Pierre e Houéto (2002) evita os pontos de mínimos locais aceitando soluções ocasionais que não melhorem o valor da função-objetivo, mas que podem levar a soluções melhores posteriormente. Dessa forma, durante a geração do conjunto V de candidatas a soluções presentes em uma vizinhança da solução corrente, removem-se as soluções que estejam presentes numa lista T, chamadas de soluções tabus. Assim, cada solução intermediária x_{i+1} é obtida resolvendo-se o problema de otimização:

$$f(x_{i+1}) = \min_{x_i \in V-T} f(x_i) \tag{2.62}$$

onde a vizinhança V depende da solução corrente x_i. O algoritmo pára quando em um número k_{max} de iterações nenhuma melhora ocorreu, ou quando todas as candidatas a solução na vizinhança são tabus, ou seja, V - T = ∅. Considerando que:

- s* é a melhor solução corrente;
- nbiter é o contador de iterações;
- bestiter é a iteração na qual foi encontrada a melhor solução;
- f é a função-objetivo;
- N(s) é o conjunto de todas soluções possíveis na vizinhança de s.

O algoritmo de BT proposto por Pierre e Houéto (2002) é dado por:

1) Escolha uma solução inicial s.

   Faça: s* := s; nbiter := 0; bestiter = 0; T = ∅; continue = TRUE;

2) Enquanto (continue = TRUE) faça:

   Se (nbiter – bestiter < k_{max}) & (V – T ≠ ∅)

   Então

   nbiter := nbiter + 1;

   gere V ⊆ N(s);

   Encontre s' em V, tal que f(s')= min f(s), ∀ s ∈ V - T;

   s : = s';

   Atualize T;

   Se f(s') < f(s*)

   Então

   s* := s';

   bestiter := nbiter;

   Fim_Se

   Fim_Se

Fim_Enquanto

O algoritmo faz uso de três estruturas de memória (curto prazo, médio prazo e longo prazo) descritas a seguir. A memória de curto prazo consiste de informações dos movimentos já realizados que levam uma solução à outra, usada com a finalidade de evitar que nenhuma das k últimas soluções visitadas seja considerada novamente como solução. Este algoritmo parte de uma solução inicial obtida através da simples atribuição de cada célula ao comutador mais próximo. O objetivo da memória de curto prazo é melhorar a solução corrente através da diminuição de seu custo ou através da diminuição das penalidades.

Seja o movimento de re-atribuição da célula a ao comutador b, denotado por m(a,b), que não considera as restrições de capacidade, e seja a vizinhança $N_{m(a,b)}(S)$ de uma solução S, definida por todas as soluções possíveis obtidas por um movimento m(a,b) em S.

Para avaliar as soluções presentes em $N_{m(a,b)}(S)$, seja o ganho $G_S(a,b)$ respectivo ao movimento m(a,b) em S, definido por:

$$G_S(a,b) = \begin{cases} \sum_{i=1,i\neq a}^{n} (h_{ai} + h_{ia}) x_{ib_0} - \sum_{i=1,i\neq a}^{n} (h_{ai} + h_{ia}) x_{ib} + c_{ab} - c_{ab_0} \text{ se } b \neq b_0 \\ M, \text{ caso contrário} \end{cases} \quad (2.63)$$

onde:

- b_0 é o comutador ao qual a célula s está atribuída, e
- M é um número arbitrariamente grande.

O algoritmo seleciona o movimento com o pior ganho entre todos os movimentos possíveis. Se $b \neq b_0$, $G_S(a,b)$ representa a subtração que incide sobre o custo f(S) da solução S ao realizar o movimento m(a,b). No caso em que $b = b_0$ não há alteração sobre os custos, ou seja, $G_S(a,b)$ é nulo. Neste caso, atribui-se um valor M arbitrariamente grande para que, ao encontrar um mínimo local com nenhum movimento possível na solução corrente S, não haja ciclos provenientes do movimento $m(a,b_0)$ que tenha o ganho mínimo. O custo da nova solução S' é simplesmente obtido por:

$$F(S') = f(S) + G_S(a,b) \quad (2.64)$$

Inicialmente, geram-se todos os ganhos $G_S(i,k)$ em uma tabela $n \times m$, onde n refere-se ao número de células e m o número de comutadores. Após cada movimento m(a,b), atualizam-se os ganhos da tabela. A atualização é simples, pois estão incluídas nos cálculos apenas as colunas relativas aos comutadores b e b_0 e as linhas correspondentes à célula a e às células atribuídas ao comutador b ou b_0.

Seja C_p o comutador da célula p na nova solução S'. Os cálculos dos ganhos $G_{S'}(p,q)$ são dados por:

$$G_{S'}(p,q) = G_S(p,q) + h_{ap} + h_{pa} \quad \text{se} \begin{cases} C_p = b, p \neq a, q \neq b_0, q \neq b \\ \text{ou} \\ C_p \neq b, C_p \neq b_0, q \neq b \end{cases} \tag{2.65}$$

$$G_{S'}(p,q) = G_S(p,q) - (h_{ap} + h_{pa}) \quad \text{se} \begin{cases} C_p = b_0, q \neq b_0, q \neq b \\ \text{ou} \\ C_p \neq b, C_p \neq b_0, q = b \end{cases} \tag{2.66}$$

$$G_{S'}(p,q) = G_S(p,q) - 2(h_{ap} + h_{pa}) \quad \text{se } C_p = b_0, q = b \tag{2.67}$$

$$G_{S'}(p,q) = G_S(p,q) + 2(h_{ap} + h_{pa}) \quad \text{se } C_p = b, p \neq a, q \neq b_0 \tag{2.68}$$

$$G_{S'}(a,b) = M \tag{2.69}$$

$$G_{S'}(a,b) = -G_S(a,b) \tag{2.70}$$

$$G_{S'}(a,q) = G_S(a,q) - G_S(a,b) \quad \text{se } q \neq b_0, q \neq b \tag{2.71}$$

$$G_{S'}(p,q) = G_S(p,q) \quad \text{para os demais casos.} \tag{2.72}$$

Em cada iteração do algoritmo define-se a vizinhança da solução corrente gerando todos os movimentos m(a,b) possíveis e escolhendo a solução resultante com pior ganho. Se não existir nenhum movimento que melhore o custo, a solução que gerar o mínimo de perda é selecionada. Para cada movimento escolhido, o método mantém na lista tabu o movimento inverso m(a,b'), onde b' denota o comutador em que a célula estava atribuída antes de aplicar o movimento m(a,b). O algoritmo pára se k_{max} iterações foram realizadas desde a última melhor solução ou se não existe nenhum movimento possível.

Pode-se dizer que existe mais chance de encontrar uma solução melhor com o aumento no número de iterações k_{max}. No entanto, se k_{max} for muito grande a busca também pode se afastar das soluções factíveis e ter pouca chance de retornar às factíveis. O algoritmo de Pierre e Houéto (2002) faz uso de um mecanismo chamado *callback* para assegurar o retorno à exploração de soluções factíveis, em que após um número de consecutivas soluções infactíveis, penalizam-se incrementalmente os ganhos relativos aos

movimentos que implicam em soluções não factíveis, ou seja, adicionando um valor periodicamente. Se os ganhos de movimentos adicionam uma célula a um comutador que tenha uma capacidade residual negativa, adiciona-se um valor composto de uma parte constante e outra variável sobre os ganhos de movimentos. A parte constante é composta do ganho do movimento e a variável consiste da introdução de um multiplicador sobre a parte variável. A cada solução infactível, este multiplicador é incrementado até um valor máximo Max_{serv}. Na primeira solução factível encontrada, o mecanismo *callback* é desativado. Dessa forma pretende-se evitar que boas soluções sejam negligenciadas.

Embora a BT procure evitar ciclos, às vezes pode ser vantajoso retornar a uma solução tabu e tomar uma outra direção diferente da escolhida anteriormente. Para obter este resultado é usada a memória de médio prazo. O componente de memória de médio prazo tem o objetivo de intensificar a busca local em regiões consideradas promissoras. No entanto, definir uma região promissora não é tarefa fácil. Pierre e Houéto (2002) usaram uma lista com as últimas melhores soluções e seus ganhos respectivos. Toda vez que surge uma solução s' melhor que a última encontrada, esta passa a fazer parte da lista. A idéia é verificar se boas soluções não estão longe umas das outras. Se elas estiverem próximas então uma região que as contenha pode ser considerada promissora e a busca deve ser intensificada nesta região.

A intensificação deve variar os movimentos em direções negligenciadas pelos movimentos da memória de curto prazo. São sugeridos dois tipos de movimentos de intensificação:

- *Movimento* $i_1(a,b)$, que permuta as duas células a e b com piores ganhos. O objetivo deste movimento é melhorar a avaliação de uma solução através do decréscimo do custo associado. Reatribui-se as duas células com piores ganhos em uma tabela, sem considerar o fato de que após a remoção de a, a tabela de ganhos é alterada e a célula b pode não ser necessariamente a melhor para ser removida.

- *Movimento $i_2(a,b)$,* que reatribui a célula a ao comutador b para ordenar o reestabelecimento das restrições de capacidade. O objetivo deste movimento é restaurar as restrições de capacidade e reduzir os cálculos da solução e das penalidades. Este movimento consiste em:

  - determinar o comutador c' que tem a capacidade residual mínima;
  - encontrar a célula a atribuída a c' que gera o volume mínimo de chamadas;
  - atribuir a célula a ao comutador b que tenha uma capacidade residual suficiente e que resulte no ganho mínimo.

Este movimento baseia-se nas restrições de capacidade e ganho através da construção de uma lista tabu do mesmo tamanho da formada pelo componente de memória de curto prazo sem nenhum critério de otimização.

O componente de memória de médio prazo, assim como o critério de memória de curto prazo pára se a última melhor solução não tem sido trocada nas últimas k_{Max} iterações ou, se não existem mais movimentos possíveis. Neste caso prossegue-se com um mecanismo de diversificação através de um componente de memória de longo prazo.

Para diversificar a busca, uma estrutura de memória de longo prazo é usada visando regiões não exploradas. Para isto, deve-se gerar novas soluções iniciais e reiniciar o processo de busca. No algoritmo de Pierre e Houéto (2002), utiliza-se uma matriz $F_{n\times m}$, cujos elementos F(a, b) armazenam o número de vezes que a antena celular a está atribuída ao comutador b nas soluções já visitadas. Gera-se então uma nova solução inicial em que para cada célula a, escolhe-se o comutador ao qual ela foi menos atribuída e a partir desta solução inicial reinicia-se o processo de busca.

Uma outra abordagem de BT para o PAACC é apresentada por Quintero e Pierre (2003). Este algoritmo pode ser descrito como: Sejam X_i a solução corrente, $N(X_i)$ todas as soluções possíveis da vizinhança de X_i e Y é uma solução factível da vizinhança de X_i. Se Y é obtido a partir de X_i através de um único movimento fora da lista tabu, e Y não pertence à lista tabu então troque X_i com o melhor ponto em $N(X_i)$. Pare depois de

um número máximo de passos ou se $N(X_i)$ está vazio. O algoritmo de Quintero e Pierre (2003) também utiliza mecanismos de intensificação e diversificação de busca para executar uma exploração mais extensiva de regiões mais promissoras que possam levar a um ponto de ótimo e privilegie regiões ainda não visitadas, evitando assim permanecer em pontos de mínimos locais.

2.7 Uma Abordagem de Algoritmo Genético Paralelo para o PAACC

Algoritmos Genéticos (AGs) são baseados no conceito de seleção natural, em que os indivíduos mais adaptados têm mais chance de sobreviver e gerar descendência. Nos AGs parte-se de populações iniciais de soluções e através de mecanismos de seleção natural, cruzamento e mutação genética evolui-se para populações compostas de soluções melhores até que se obtenha uma solução que satisfaça uma dada condição de otimalidade. AGs são compostos de três fases básicas: *criação* de uma população inicial, *alteração* da população usando operadores genéticos sobre os elementos deste conjunto e *avaliação* da população. A alteração é realizada por meio de vários operadores genéticos, tais como seleção de elementos da população, cruzamento entre eles, mutação e migração entre populações. Espera-se que cada nova geração seja melhor que as gerações anteriores.

Uma abordagem de algoritmo genético para o PAACC é apresentada por Quintero e Pierre (2002). Nesta abordagem, os elementos da população (*cromossomos)* são representados por vetores de tamanho n. Cada elemento do vetor é chamado *gene* e representa uma antena celular. O conteúdo armazenado pelo gene é o número do comutador ao qual a antena está atribuída. No exemplo mostrado na FIGURA 2.5, tem-se que a célula 1 está atribuída ao comutador 3, a célula 2 está atribuída ao comutador 1, e assim por diante. Desta forma, os valores dos genes são inteiros e não podem exceder m.

1	2	3	...	n-2	n-1	N
3	1	m	...	3	m-1	2

FIGURA 2.5 – Exemplo de Cromossomo

Nesta representação, a restrição de que uma antena celular seja atribuída a exatamente um comutador é satisfeita por construção. No entanto, as restrições de capacidade do comutador ainda permanecem em aberto.

Para o processo de cruzamento entre um par de elementos de uma população de cromossomos, escolhe-se, aleatoriamente, um número p entre 1 e n-1. Dois novos cromossomos são criados trocando-se os valores das posições que ficam entre (p+1) e n, conforme mostra a FIGURA 2.6.

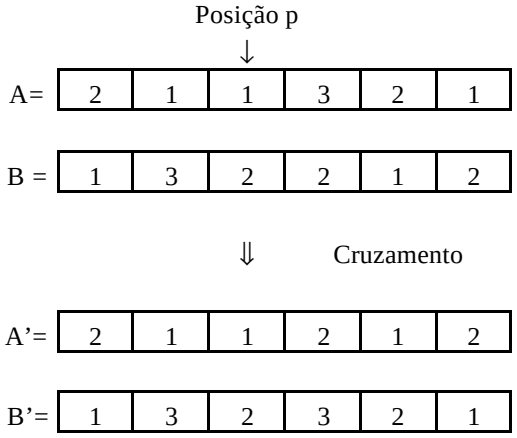

FIGURA 2.6 – Exemplo de Cruzamento

Para evitar que a população fique sujeita a cromossomos muito semelhantes, aplica-se o processo de mutação, onde um gene é escolhido aleatoriamente em um cromossomo e mudado deliberadamente, conforme ilustra a FIGURA 2.7.

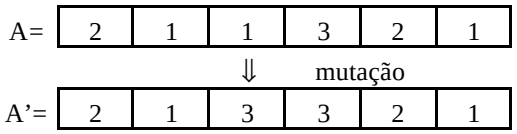

FIGURA 2.7 – Exemplo de Mutação.

A próxima geração de cromossomos é gerada por um processo de seleção e reprodução. No processo de seleção, a escolha pode ser feita probabilisticamente ou através de um critério de avaliação que determine os elementos mais saudáveis da população para se reproduzirem.

A abordagem paralela do AG consiste em particionar a população em várias subpopulações e fazer com que cada população evolua de maneira independente das outras. No entanto, de tempos em tempos permite-se a troca de características genéticas entre diferentes subpopulações através da migração.

Segundo Quintero e Pierre (2003) a eficiência e exatidão da migração dependem, dentre outros fatores:

- do número e do tamanho das populações;
- da taxa de migração, que corresponde ao número de indivíduos que deverão migrar;
- do intervalo de migração, que é o número de gerações entre cada migração; e
- dos destinos dos migrantes.

O algoritmo genético paralelo proposto é resumido a seguir:

1) Construa a população inicial POP_k.
2) Adicione POP_k ao conjunto de populações POP(gen).
3) Aplique cruzamento e mutação em POP(gen), obtendo CRIA.
4) Faça:

   POP_{k+1} = seleção em ($POP_k \cup$ CRIA).

5) Se for o momento de realizar migração então:

   Aplique seleção em (POP_{k+1}) e obtenha emigrantes;

   Determine os destinos dos emigrantes;

   Se existem imigrantes destinados à população, receba imigrantes e obtenha IMIGR, senão faça IMIGR = ϕ.

6) Faça:

POP_{k+1} = seleção em ($POP_k \cup$ IMIGR);

k = k + 1.

7) Se o critério de parada não for satisfeito, retorne ao passo 2.

2.8 Uma Abordagem de *Simulated Annealing* para o PAACC

Simulated Annealing (SA) é uma metaheurística inspirada no trabalho de Metropolis *et al*. (1953) sobre o recozimento de sólidos. Kirkpatrick *et al*. (1983) introduziram a analogia de SA em Otimização Combinatória e Cerny (1985) aprimorou a idéia. O termo *annealing* refere-se a um processo de resfriamento térmico que começa pela liquidificação de um cristal sujeito a uma alta temperatura, seguido por uma lenta e gradativa diminuição de sua temperatura até que o ponto de solidificação seja alcançado, quando então o sistema atinge um estado de energia mínima (Ignácio *et al*., 2000).

Em problemas de Otimização Combinatória, a técnica de SA permite que sejam aceitas soluções que piorem a função-objetivo. Posteriormente, estas soluções poderão ser descartadas se não trouxerem melhorias ao problema. A técnica usa aleatoriedade para decidir se aceita ou se rejeita uma solução que deteriore o custo. As propriedades estocásticas da metaheurística SA previnem que durante a busca por uma solução, o algoritmo permaneça ao redor de um mínimo local. Por exemplo, no algoritmo guloso tradicional, a qualidade do resultado final depende quase totalmente da solução inicial. Na metaheurística SA, a idéia é explorar por completo o espaço de solução de modo que a solução final se torne insensível ao estado inicial.

Na proposta de SA para o PAACC apresentada por Quintero e Pierre (2003), uma solução inicial factível é gerada e fixada como solução corrente. Uma solução da vizinhança da solução vigente é escolhida aleatoriamente. Se o custo desta solução escolhida é menor do que a solução anterior, então ela é tomada como solução corrente.

Caso contrário, ela é aceita de acordo com uma probabilidade que é calculada de acordo com o estágio do algoritmo. Este estágio é uma analogia à temperatura do processo de recozimento original.

2.9 Uma Abordagem Mista para o PAACC

Algoritmos combinando técnicas de solução também já foram propostas para o PAACC. Na abordagem proposta por Menon e Gupta (2004) a técnica de Geração de Colunas é combinada com a metaheurística SA para a solução do problema. Nesta abordagem, o PAACC é tratado como um problema de particionamento e resolvido por meio de uma relaxação de PL. Desta forma, uma solução para o PAACC pode ser vista como uma partição P do conjunto N de células em p subconjuntos disjuntos que correspondem às atribuições factíveis dos comutadores, ou seja, que não viole as restrições de capacidade dos comutadores. Se a todos os comutadores existem antenas celulares atribuídas, então p = m (caso contrário, p < m).

Nesta abordagem, definem-se os parâmetros a_{is}, b_{is} e r_s como:

- $a_{is} = \begin{cases} 1, \text{ se a célula } i \text{ está representada no subconjunto } s \in P \\ 0, \text{ caso contrário.} \end{cases}$;

- $b_{ks} = \begin{cases} 1, \text{ se o subconjunto } s \in P \text{ representa a atribuição do switch } k \\ 0, \text{ caso contrário.} \end{cases}$

- r_s = custo total de cabeamento e de *handoff* relativo a atribuição s de P;

Define-se também a variável de decisão v_s como:

- $v_s = \begin{cases} 1, \text{ se o subconjunto } s \in P \text{ é selecionado como atribuição} \\ 0, \text{ caso contrário} \end{cases}$,

Com isto, o PAACC pode ser formulado como o seguinte problema de particionamento do conjunto de células:

$$\min \quad \sum_{s \in P} r_s v_s \tag{2.73}$$

sujeito a:

$$\sum_{s \in P} a_{is} v_s = 1 \quad \text{para } i \in N \tag{2.74}$$

$$\sum_{s \in P} b_{ks} v_s \leq 1 \quad \text{para } k \in M \tag{2.75}$$

$$v_s \in \{0,1\} \quad \text{para } s \in P \tag{2.76}$$

As restrições (2.74) representam a impossibilidade de uma célula estar atribuída a mais de um comutador, as restrições (2.75) permitem que cada comutador k esteja atribuído a, no máximo, uma antena celular e as restrições (2.76) referem-se à integralidade das variáveis de decisão.

O conjunto de partições factíveis aumenta exponencialmente em relação ao número de células tornando impraticável, para problemas com muitas células, a verificação de todas possibilidades. Na abordagem proposta por Menon e Gupta (2004) considera-se um conjunto inicial $P' \subseteq P$ e resolve-se a relaxação linear do problema mestre restrito (ou seja, considerando-se $v_s \in [0,1]$). Os custos duais obtidos da resolução do problema mestre restrito são então usados na escolha das colunas que podem beneficiar o valor da função objetivo.

Sejam π_i , $i \in N$, as variáveis duais associadas às restrições (2.74) e $\mu_k \leq 0$, $k \in M$, as variáveis duais associadas às restrições (2.75). O custo reduzido associado à coluna s é representado por:

$$CR_s = \left(r_s - \sum_{i=1}^{n} \pi_i a_{is} - \mu_k \right) \tag{2.77}$$

onde:

– $r_s = \sum_{i=1}^{n}\left(c_{ik} + \sum_{j=1}^{n} h_{ij}\right)a_{is} - \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}a_{is}a_{js}$ é o custo associado à atribuição s, e

– k é o comutador associado à atribuição s.

No custo associado à atribuição s estão incluídos os custos de cabeamento $\left(\sum_{i=1}^{n} c_{ik}a_{is}\right)$ e os custos de *handoff* $\left(\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}a_{is}\right)$. Observe que os custos de *handoff* são anulados se duas células, i e j, estão atribuídas ao mesmo comutador na atribuição s $\left(-\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}a_{is}a_{js}\right)$.

Uma coluna s pode melhorar a função-objetivo do problema mestre restrito se seu custo reduzido for negativo, ou seja, se $CR_s < 0$. Portanto uma coluna vantajosa para o problema mestre restrito existe, se e somente se, o subproblema:

$$\min \quad \sum_{i=1}^{n}\left(\left(c_{ik} + \sum_{j=1}^{n} h_{ij}\right) - \pi_i\right)z_i - \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}z_i z_j - \mu_k \tag{2.78}$$

sujeito a:

$$\sum_{i=1}^{n} \lambda_i z_i \leq M_k \tag{2.79}$$

$$z_i \in \{0,1\}, i \in N \tag{2.80}$$

tem solução com valor negativo da função-objetivo. Este subproblema, no entanto, é um problema difícil de ser resolvido, pois se trata de um problema da mochila quadrático binário. Existem algoritmos próprios para este tipo de problema, como o proposto por Caprara *et al.* (2001). Na abordagem de Menon e Gupta (2004) este problema é resolvido com o uso de heurísticas.

A abordagem proposta incorpora idéias da técnica de *Simulated Annealing* ao permitir que colunas improdutivas do ponto de vista de beneficiar a função-objetivo da relaxação de PL (ou seja, colunas com custo reduzido positivo) possam ser incluídas no problema mestre restrito, visando obter boas soluções factíveis para o problema. Apresenta-se a seguir o algoritmo proposto por Menon e Gupta (2004):

1. Identificar uma solução inicial s com custo v(s);
2. Inicializar o problema mestre restrito (PMR);
3. Selecionar uma temperatura inicial $t = t_0 > 0$;
4. Repetir até que um critério de parada seja satisfeito:
    a) Resolver o PMR;
    b) Recuperar as variáveis duais;
    c) Identificar uma nova solução s' com custo v(s'). Seja s'_k uma atribuição para o comutador k em s';
    d) Calcular $\Delta s = v(s') - v(s)$; r = random(0, 1);
    e) Calcular os custos reduzidos $cr(s'_k)$ para todos os comutadores k;
    f) Se $\Delta s < 0$ então:

        Aceitar s', ou seja, fazer s = s';

        Acrescentar as atribuições s'_k em s' ao PMR;

    Senão, se $e^{\Delta s / t} \geq r$ então:

        Aceitar s', ou seja, fazer s = s';

        Acrescentar as atribuições s'_k em s' ao PMR;

Senão, se cr(s'_k) < 0 para algum comutador k, então:

Acrescentar s'_k ao PMR;

Senão, se $e^{cr(s'_k)/t} \geq r$, então:

Acrescentar s'_k ao PMR;

Senão, rejeitar s'_k.

Fim_Se

g) Reduzir a temperatura apropriadamente;

Fim_Repetir.

Neste capítulo foram revisados alguns dos principais textos que tratam do PGA e do PAACC. No capítulo a seguir serão apresentados os métodos de resolução para estes problemas propostos neste trabalho.

CAPÍTULO 3

OS MÉTODOS DE GERAÇÃO DE COLUNAS PROPOSTOS

Este capítulo apresenta os princípios teóricos sobre os quais se baseiam os métodos propostos neste trabalho. O capítulo inicia com a apresentação da decomposição Dantzig e Wolfe (1960) para problemas de Programação Linear inteiros e, em seguida, apresenta o método básico de Geração de Colunas, o Problema de Particionamento de Conjuntos e algumas formas de se obter limitantes através de relaxações lagrangeanas, assim como as relações existentes entre os limitantes lagrangeanos e o limite de Farley (Farley, 1990; Lorena *et al*., 2003). O capítulo apresenta ainda o Método de Cortes de Kelley (Kelley, 1960), que é uma técnica de Geração de Linhas (Neame, 1999), e uma proposta de Ryan e Foster (1981) de como ramificar a árvore de busca de um método B&B para encontrar soluções exatas de problemas de particionamento de conjuntos. Ao final, este capítulo apresenta os métodos propostos para o PGA e para o PAACC, que compreendem a técnica de Geração de Colunas combinada com a relaxação lagrangeana/*surrogate*. Apresenta também, para o PGA, como o método proposto pode ser combinado com um algoritmo *Branch-and-Bound* e constituir um método *Branch-and-Price* para obter soluções exatas para o problema de atribuição.

3.1 A Decomposição de Dantzig-Wolfe

Seja um problema de Programação Linear inteiro da forma:

$$\max \quad c\,x \tag{3.1}$$

sujeito a:

$$A\,x = b \tag{3.2}$$

$$A'x \leq b' \tag{3.3}$$

$$x \text{ inteiro} \tag{3.4}$$

onde: A (uma matriz $m \times n$), A' (uma matriz $m' \times n$), c (um vetor $n \times 1$), b (um vetor $m \times 1$) e b' (um vetor $m' \times 1$) são conhecidos, e a variável de decisão x é um vetor $n \times 1$.

A decomposição de Dantzig-Wolfe para problemas deste tipo é recomendada quando as restrições (3.2) e (3.3) são tais que podem ser trabalhadas separadamente. Estas restrições são, então, chamadas de restrições gerais e restrições especiais, respectivamente. Em geral, as restrições especiais têm uma interpretação natural (restrições da mochila, por exemplo).

Definindo $S = \{ x \text{ inteiro} / A'x \leq b' \}$, tem-se que se $\{ x^k / k \in K \}$ é o conjunto dos pontos extremos de S, sendo K o conjunto de índices destes pontos, e se S é limitado e fechado, então $x \in S$ pode ser representado como uma combinação convexa linear de pontos extremos de S, ou seja:

$$x = \sum_{k \in K} \alpha_k x^k \tag{3.5}$$

onde:

$$\sum_{k \in K} \alpha_k = 1 \quad \text{e} \quad \alpha_k \geq 0, \, k \in K \tag{3.6}$$

Com isto, o problema original (3.1)-(3.4) pode ser escrito como:

$$\max \quad c \sum_{k \in K} \alpha_k x^k \tag{3.7}$$

sujeito a:

$$A\left(\sum_{k \in K} \alpha_k x^k \right) = b \tag{3.8}$$

$$\sum_{k \in K} \alpha_k = 1 \tag{3.9}$$

$$\alpha_k \geq 0, \, k \in K \tag{3.10}$$

Assim, o problema de encontrar soluções viáveis (extremos) para S é chamado de *subproblema* e o problema (3.7)-(3.10) é denominado como *problema-mestre*, e esses dois problemas (subproblema e problema-mestre) definem a decomposição de Dantzig-Wolfe.

3.2 O Método de Geração de Colunas

Em geral, o número de pontos extremos x^k é grande e, por razões práticas, opta-se por trabalhar apenas com um número restrito de extremos. Seja $K^\ell \subseteq K$ um subconjunto que representa os índices de pontos extremos gerados até a iteração ℓ. O problema-mestre (3.7)-(3.10) quando se considera $K = K^\ell$, ou seja, quando se considera apenas um subconjunto dos extremos de S, passa a ser denominado *problema-mestre restrito.* Neste caso, o subproblema pode ser chamado de *subproblema de geração de colunas*, pois a variáveis obtidas pela resolução deste problema constituirão novas colunas para o problema-mestre restrito. A fim de gerar variáveis que melhorem a função-objetivo do problema-mestre restrito, os índices destas variáveis são determinados pelo cálculo dos custos reduzidos:

$$z_k - \hat{c}_k = \max_{1 \le j \le s} (c - \lambda A) x^j - \mu \qquad (3.11)$$

em que λ e μ são as variáveis duais do problema-mestre restrito referentes às restrições (3.8) e (3.9), respectivamente, e x^j (j = 1, ..., s) os extremos de S. Desta forma, o subproblema torna-se um *problema de custo reduzido*, também conhecido como *problema pricing*.

Assim, se $z_j - c_j \le 0$, $\forall\, j \in \{1, ..., s\}$, então a solução é ótima. Se existir algum $\hat{k}$ tal que $z_{\hat{k}} - \hat{c}_{\hat{k}} > 0$, então a variável $\alpha_{\hat{k}}$ é candidata a fazer parte da solução, ou seja, $K^{\ell+1} = K^\ell \cup \{\hat{k}\}$.

Como S é poliédrico e limitado, e x^j, j = 1, ..., s são os extremos de S, então a solução do problema linear será um dos extremos de S, ou seja, a solução ótima de (3.11) é equivalente à solução ótima do problema dado por:

$$\max_{x \in S} (c - \lambda A)x - \mu \tag{3.12}$$

Assim, o algoritmo básico de geração de colunas pode ser estabelecido como:

1) Determine um conjunto inicial K^0 de colunas para o problema-mestre restrito. Faça $\ell = 0$;

2) Resolva a relaxação de Programação Linear do problema-mestre restrito;

3) Utilize os valores duais ótimos obtidos no passo anterior e resolva o subproblema para obter novas colunas $x^{\hat{k}}$ em *S*;

4) Adicione as colunas $x^{\hat{k}}$ com custos reduzidos positivos à formulação do problema-mestre restrito. Se não houver tais colunas, então pare, pois a solução é ótima;

5) Caso contrário, faça $K^\ell \leftarrow K^\ell \cup \{\hat{k}\}$, $\ell \leftarrow \ell + 1$ e volte para o passo 2.

Uma vez as evitadas soluções cíclicas no problema-mestre e a degeneração no subproblema, o algoritmo converge em um número finito de iterações, pois consiste do algoritmo Simplex Revisado com a inclusão dos custos reduzidos dados pela resolução do subproblema. O subproblema não precisa ser completamente otimizado em cada iteração. É necessário apenas que as colunas geradas x^k tenham custos reduzidos positivos para que seus respectivos λ_k sejam candidatos a entrar na base.

3.3 O Problema de Particionamento de Conjuntos

O problema de determinar uma partição de um conjunto de modo a satisfazer algumas restrições e otimizar uma função linear é conhecido como Problema de Particionamento

de Conjuntos (PPC). Como já observaram Barnhart *et al*. (1998), muitos métodos de solução para problemas de atribuição já foram desenvolvidos levando-se em conta formulações baseadas no PPC.

Sejam:

- $M = \{1, ..., m\}$, um conjunto finito;
- $P = \{P_1, ..., P_n\}$ o conjunto de todos os subconjuntos de M; e
- $N = \{1, 2, ..., n\}$, o conjunto de índices de P.

Um subconjunto F de N ($F \subseteq N$) é uma partição de M se $\bigcup_{j \in F} P_j = M$ e $P_j \cap P_k = \phi$, $\forall$ j, $k \in F$.

Se o subconjunto P_j ($j \in N$) é representado pelo vetor $z^j = (z_{1j}, \cdots, z_{mj})^t$, tal que:

$$z_{ij} = \begin{cases} 1, \text{ se o elemento } i \text{ está no conjunto } P_j \\ 0, \text{ caso contrário} \end{cases}, \quad \forall i \in M \tag{3.13}$$

então, o PPC pode ser formulado como:

$$\max \sum_{j \in N} c\left(z^j\right) \tag{3.14}$$

sujeito a:

$$\sum_{j \in N} z_{ij} = 1 \qquad \forall i \in M \tag{3.15}$$

$$z^j \in P \tag{3.16}$$

onde $c(z^j)$ é o custo associado ao subconjunto P_j.

Em geral, os subconjuntos P_j factíveis de um problema são tais que:

$$P_j = \left\{ z^j / A_j z^j \leq b_j \right\} \qquad j \in N \tag{3.17}$$

ou seja, para cada subconjunto existem diferentes restrições a serem satisfeitas. Este é o caso dos problemas de atribuição considerados neste trabalho, uma vez que para estes problemas cada subproblema refere-se a um agente, e as características dos agentes são diferentes.

3.4 A Relaxação Lagrangeana

Aplicando a relaxação lagrangeana nas restrições (3.2) do problema (3.1)-(3.4), obtém-se o seguinte problema:

$$\max_{x \in S} c\,x + \lambda\,(b - A\,x) \tag{3.18}$$

sujeito a (3.3) e (3.4).

Para quaisquer λ e x factível, sabe-se que $[\max_{x \in S} c\,x + \lambda\,(b - A\,x)]$ é um limite superior para o problema inteiro original. O dual lagrangeano, dado por:

$$\min_{\lambda} \max_{x \in S} \{\, c\,x + \lambda\,(b - A\,x) \,\} \tag{3.19}$$

é um limitante mais refinado e na pior das hipóteses se iguala ao limitante de relaxação linear do problema inteiro.

Considerando que $\{ x^k / k \in K \}$ é o conjunto dos pontos extremos de S, o dual lagrangeano pode reescrito como:

$$\min_{\lambda} \max_{x \in S} \{\, c\,x + \lambda\,(b - A\,x) \,\} =$$

$$= \min_{\lambda} \{ b\,\lambda + \max_{k \in K} \{\, (c - \lambda\,A)\,x^k \,\} \tag{3.20}$$

Com isto, o problema dual lagrangeano é equivalente ao seguinte problema:

$$\min b\,\lambda + \mu \tag{3.21}$$

sujeito a:

$$\mu \geq c\,x^k - \lambda\,A\,x^k \tag{3.22}$$

que corresponde ao dual do problema-mestre (3.7)-(3.10).

3.5 O Método de Cortes de Kelley

Seja $x^{\hat{k}}$ a variável gerada pelo subproblema (3.12) numa dada iteração ℓ do algoritmo de geração de colunas e $(\hat{\lambda}, \hat{\mu})$ as variáveis duais do problema-mestre restrito. Se $(\hat{\lambda}A - c)x^{\hat{k}} + \mu \leq 0$, ou seja, se $(c - \hat{\lambda}A)x^{\hat{k}} \geq \mu$, a solução é ótima e o conjunto K^{ℓ} possui uma base ótima para o problema original. Caso contrário, $\hat{k}$ é incluído no conjunto de índices K corrente e a restrição $\mu \geq c\ x^k - \lambda\ A\ x^k$ é acrescentada ao problema-mestre dual (3.21)-(3.22). Este procedimento é conhecido como de método de cortes de Kelley. Este método é também conhecido como geração de linhas, em analogia à técnica de geração de colunas.

3.6 A Relaxação Lagrangeana/*Surrogate*

A relaxação lagrangeana/*surrogate* consiste em aplicar a relaxação lagrangeana sobre restrições já relaxadas pela relaxação *surrogate.* Aplicando a relaxação *surrogate* sobre as restrições (3.2) do problema (3.1)-(3.4), tem-se o seguinte problema:

$$\max_{x \in S} \quad c\,x \tag{3.23}$$

sujeito a:

$$\lambda\,A\,x = \lambda\,b \tag{3.24}$$

onde $\lambda = (\lambda_1, ..., \lambda_m)$ é o vetor formado pelos multiplicadores *surrogate* e que transformaram a restrição (3.2) numa única restrição $\lambda A x = \lambda b$. Seja t o multiplicador lagrangeano aplicado sobre esta restrição. Tem-se a relaxação lagrangeana/*surrogate* do problema:

$$\max_{x \in S} \quad c x + t \lambda (b - A x) \tag{3.25}$$

Se t = 1, a relaxação lagrangeana/*surrogate* é idêntica à relaxação lagrangeana. Com um valor conveniente para t, denominado de multiplicador lagrangeano/*surrogate*, pode-se melhorar o limitante produzido pela relaxação. A busca pelo melhor multiplicador t corresponde a resolver o problema dual dado por:

$$\min_{t \geq 0} \max_{x \in S} \quad c x + t \lambda (b - A x) \tag{3.26}$$

A determinação de novas colunas através da resolução do subproblema gerador de colunas (3.12) pode ser modificada pelo multiplicador lagrangeano/*surrogate*. Neste caso, o novo subproblema será dado por:

$$\max_{x \in S} \{cx - t\lambda Ax\} \tag{3.27}$$

Para *t* ? 1, os problemas (3.12) e (3.27) podem produzir colunas diferentes para o método de Geração de Colunas. Se as colunas x^p obtidas em (3.27) satisfizerem a condição $cx^p - \lambda Ax^p - \mu \geq 0$, estas colunas poderão ser acrescentadas ao problema-mestre restrito. Assim, o multiplicador lagrangeano/*surrogate* t não interfere no critério de entrada das colunas geradas no problema-mestre restrito, mas pode influenciar no número e na estrutura das colunas candidatas.

Para a determinação do multiplicador t, seja $v(SP^{\lambda})$, o valor da solução do problema (3.23)-(3.24) e $v(L_t SP^{\lambda})$, o valor da solução da relaxação lagrangeana/*surrogate* (3.25). Considerando um multiplicador λ fixo, o melhor valor para o fator lagrangeano/*surrogate* t pode ser obtido resolvendo-se o dual (3.26), ou seja,

resolvendo-se $v(D_t^{\lambda}) = \min_{t \geq 0} c\,x + t\,\lambda(b - Ax)$. Portanto, tem-se a seguinte relação de limitantes:

$$v(D_t^{\lambda}) \leq v(SP^{\lambda}) \leq v(L_{t=1} SP^{\lambda}) \tag{3.28}$$

Sabe-se que a função lagrangeana $L(t) = \max_{x \in S} \{ c\,x + t\,\lambda\,(b - A\,x)\}$ é linear por partes, convexa e não diferenciável (Nemhauser e Wolsey, 1988). O melhor valor da relaxação lagrangeana/*surrogate*, que pode ser obtido realizando uma busca para diferentes valores de t, resulta num limitante melhor ou igual ao da relaxação lagrangeana usual. Contudo, sempre existe um intervalo $[t_0, t_1]$ (com $t_0 = 1$ ou $t_1 = 1$) para valores do multiplicador t que também produz limitantes melhores que os obtidos pela relaxação lagrangeana tradicional, como mostra a FIGURA 3.1 (na qual escolheu-se, arbitrariamente, $t_1 = 1$).

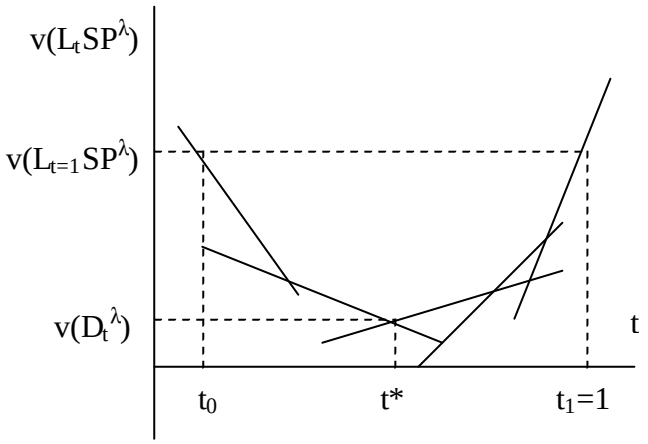

FIGURA 3.1 – Limite lagrangeano/*surrogate*

Assim, para obter um limitante melhor do que o obtido pela relaxação lagrangeana usual não é necessário encontrar o melhor valor para o multiplicador lagrangeano/*surrogate* (t*). Para isto, basta determinar um valor de t entre t_0 e t_1, o que pode ser feito por meio de uma heurística de busca, como a proposta por Senne e Lorena (2000), descrita a seguir.

Sejam:

- s, um tamanho inicial de passo;
- k, o número de iterações;
- k_{max}, o número máximo de iterações;
- t_0, um valor inicial para o multiplicador lagrangeano/*surrogate*;
- t, o valor corrente para o multiplicador lagrangeano/*surrogate*;
- T, o melhor valor encontrado para o multiplicador lagrangeano/*surrogate*;
- z, o valor mínimo para $v(L_tSP^{\lambda})$.

Com isto, o algoritmo de busca do multiplicador lagrangeano/*surrogate* pode ser estabelecido como:

Faça inicialmente: k = 0; z = 0; t = t_0; T = t; $t^+ = t^- = \Lambda$, onde o símbolo Λ indica o valor "indefinido");

Enquanto ($k < k_{max}$), repita:

k = k + 1;

Resolva (L_tSP^{λ}) e obtenha x^{λ};

Se ($v(L_tSP^{\lambda}) < z$) então, faça:

$z = v(L_tSP^{\lambda})$; T = t;

Calcule a inclinação da função L(t) como: $\Delta^{\lambda} = \lambda(b - Ax^{\lambda})$;

Se $\Delta^{\lambda} > 0$ então, fazer:

$t^- = t$; $z^- = z$;

Se ($t^+ = \Lambda$) então, fazer: t = t + s;

Senão:

Tente melhorar o multiplicador corrente resolvendo (L_tSP^{λ}) para $t = \frac{z^+t^+ + z^-t^-}{z^+ + z^-}$, atualizando T, se necessário, e pare.

Fim_Se;

Senão:

$t^+ = t$; $z^+ = z$;

Se ($t^- = \Lambda$) então, fazer: $t = t - s$;

Senão:

Tente melhorar o multiplicador corrente resolvendo (L_tSP^{λ}) para $t = \frac{z^+t^+ + z^-t^-}{z^+ + z^-}$, atualizando T, se necessário, e pare.

Fim_Se;

Fim_Se;

Senão:

Tente melhorar o multiplicador corrente resolvendo (L_tSP^{λ}) para $t = \frac{t-s}{2}$, atualizando T, se necessário, e pare.

Fim_Se;

Fim_Enquanto.

3.7 O Limite de Farley

O limite de Farley (1990) pode ser relacionado com o limite lagrangeano/*surrogate*, como será mostrado a seguir.

Sejam z_P e z_D os valores das funções-objetivo do problema-mestre restrito dado por (3.7)-(3.10) e do dual lagrangeano dado por (3.19). Como estes problemas constituem pares primal e dual, tem-se que:

$$z_P \leq z_D \tag{3.29}$$

Aplicando o multiplicador lagrangeano/*surrogate* t na expressão do custo reduzido (3.12), tem-se:

$$CR_t = \max_{x \in S} (c - t\lambda A)x - \mu \geq cx - t\lambda Ax - \mu, \forall x \in S \tag{3.30}$$

Assim:

$$c\,x - t\,\lambda\,A\,x \leq CR_t + \mu, \forall x \in S \tag{3.31}$$

Portanto, $(t\,\lambda,\ CR_t + \mu)$ é viável para o dual (3.21)-(3.22) e como vale a expressão (3.29), tem-se:

$$z_P \leq t\,\lambda\,A\,x + CR_t + \mu \tag{3.32}$$

Assim, $(t\,\lambda\,A\,x + CR_t + \mu)$ é um limitante superior para o valor da função-objetivo do problema-mestre restrito.

Farley (1990) propôs um limite para o caso particular em que $c \geq 0$ e $x \geq 0$. Fazendo $t = t_0$, tal que:

$$\max_{x \in S} \{cx - t_0 \lambda Ax\} = 0 \tag{3.33}$$

tem-se:

$$z_P \leq t_0 \lambda b \tag{3.34}$$

e a expressão ($t_0 \lambda b$) é conhecida como limite de Farley.

Pode-se estabelecer uma relação entre os limites produzidos pelas relaxações lagrangeanas e o limite de Farley, por meio do multiplicador lagrangeano/*surrogate* t da seguinte maneira:

- Se t = 1, tem-se o limite produzido pela relaxação lagrangeana tradicional;
- Se t = t*, onde t* é obtido por (3.26), tem-se o melhor limite produzido pela relaxação lagrangeana/*surrogate*;
- Se t = t_0, onde t0 é obtido por (3.33), tem-se o limite de Farley, para os casos onde $c \geq 0$ e $x \geq 0$.

Comparando os limites lagrangeano e lagrangeano/*surrogate* tem-se que:

$$\max_{x \in S} c\,x + t^* \lambda (b - A\,x) \leq \max_{x \in S} c\,x + \lambda (b - A\,x) \tag{3.35}$$

Da mesma forma, pode-se estabelecer uma relação entre o limite de Farley, quando existe, e o limite lagrangeano/*surrogate* pela seguinte expressão:

$$\max_{x \in S} c\,x + t^* \lambda (b - A\,x) \leq t_0 \lambda b \tag{3.36}$$

Assim, pode-se concluir que a relaxação lagrangeana/*surrogate* produz o melhor limitante (Lorena *et al*., 2003).

3.8 A Busca de Soluções Exatas

A menos que as soluções α_k de (3.7)-(3.10) sejam todas inteiras, será necessário aplicar um algoritmo B&B para obter a solução ótima inteira do problema. Uma decisão crítica em um algoritmo B&B é como separar o problema em subproblemas, que se traduz em como ramificar a árvore de busca. Descreve-se aqui uma estratégia proposta por Ryan e

Foster (1981), e citada por Vance *et al*. (1994), para problemas de particionamento de conjuntos. A estratégia se baseia em uma proposição para o caso em que b = 1, ou seja, quando a restrição (3.8) pode ser escrita como $A\alpha = 1$.

Se $A = (a_{ij})$ é uma matriz 0-1 e uma solução básica para $A\alpha = 1$ é fracionária, isto é, pelo menos um dos componentes de α é fracionário, então, segundo Vance *et al*. (1998), existem duas linhas r e s do problema-mestre tal que:

$$0 < \sum_{k \in C_{rs}} \alpha_k < 1 \tag{3.37}$$

onde C_{rs} é o conjunto de colunas de A nas quais $a_{rk} = 1$ e $a_{sk} = 1$.

Considerando que $\alpha_{k'}$ é uma variável fracionária e r uma linha qualquer em que $a_{rk'} = 1$, pela restrição (3.8) deve existir um valor k'' com $a_{rk''} = 1$ tal que $\alpha_{k''}$ é também fracionária. Considerando que não existem colunas duplicadas na base de solução do método Simplex, deve existir s tal que vale uma das relações: $a_{sk'} = 1$ ou $a_{sk''} = 1$, mas não ambas. Assim, para as variáveis fracionárias $\alpha_{k'}$ e $\alpha_{k''}$ pode-se identificar em A uma submatriz de ramificação como, por exemplo, a mostrada na FIGURA 3.2.

$[a_{ij}]$	j = k'	j = k''
i = r	1	1
i = s	0	1

FIGURA 3.2 – Exemplo de submatriz de ramificação

Tal submatriz de ramificação implica nas seguintes relações:

$$1 = \sum_{k} a_{rk}\alpha_k = \sum_{k \in C_r} \alpha_k > \sum_{k \in C_{rs}} \alpha_k \tag{3.38}$$

onde C_r é o conjunto das colunas de A nas quais $a_{rk} = 1$. Note que a desigualdade segue do fato de que a última somatória inclui $\alpha_{k'}$ ou $\alpha_{k''}$, mas não ambas.

Assim, o par (r, s) pode ser usado para estabelecer as seguintes restrições de ramificação:

- Ramificação esquerda: $\sum_{k \in C_{rs}} \alpha_k = 1$ (3.39)

- Ramificação direita: $\sum_{k \in C_{rs}} \alpha_k = 0$ (3.40)

Com isto, para o nó ramificado pela esquerda, as linhas r e s são consideradas em uma mesma coluna e, para o nó ramificado pela direita, as linhas r e s são consideradas em colunas diferentes.

Considerando, conforme (3.17), que para cada subconjunto factível P_j as restrições do problema são diferentes, pelo esquema sugerido por Ryan e Foster (1981), os elementos r e s devem pertencer ao mesmo subconjunto para o nó do ramo da esquerda e em subconjuntos diferentes para o nó do ramo direito da árvore. Assim, para o nó esquerdo da ramificação, deve-se ter:

- $(\alpha_{rk} = 0$ e $\alpha_{sk} = 0)$ ou $(\alpha_{rk} = 1$ e $\alpha_{sk} = 1)$ (3.41)

enquanto que para um nó direito da ramificação toda coluna factível deve ter:

- $(\alpha_{rk} = 0$ e $\alpha_{sk} = 0)$ ou $(\alpha_{rk} = 0$ e $\alpha_{sk} = 1)$ ou $(\alpha_{rk} = 1$ e $\alpha_{sk} = 0)$ (3.42)

Sejam P_s o conjunto de índices que representam todas as colunas referentes à tarefa s vigentes no PMR, α_k^s a variável de decisão associada à k-ésima coluna e x_{rk}^s o componente da coluna k na linha r. Considerando as restrições de ramificação (3.42) obtêm-se as seguintes implicações:

- $\sum_{k \in P_s} \alpha_k^s = 1 \Rightarrow \sum_{k \in P_s} x_{rk}^s \alpha_k^s = 1 \Rightarrow z_{rs} = 1$ (3.43)

e

$$- \quad \sum_{k \in P_s} \alpha_k^s = 0 \Rightarrow \sum_{k \in P_s} x_{rk}^s \alpha_k^s = 0 \Rightarrow z_{rs} = 0 \tag{3.44}$$

3.9 Os Métodos de Geração de Colunas Propostos

Os métodos de Geração de Colunas propostos neste trabalho consideram a aplicação da decomposição de Dantzig-Wolfe de forma a reformular os problemas de atribuição como problemas de cobertura de conjuntos. A fim de construir a decomposição do PGA, considere $K_i = \left\{ x_1^i, ..., x_{k_i}^i \right\}$ o conjunto de todas as possibilidades de atribuir as tarefas às máquinas i (i = 1, ..., m), onde $x_k^i = \left(x_{1k}^i, ..., x_{nk}^i \right)$ é uma solução factível para (1.12)-(1.13).

Sejam y_k^i para $i \in M = \{1, ..., m\}$ e $k \in K_i$ a variável binária que indica se o padrão de atribuição de tarefas x_k^i foi atribuído ou não à máquina i, ou seja:

$$y_k^i = \begin{cases} 1, \text{se } x_k^i \text{ é atribuído à máquina } i; \\ 0, \text{caso contrário.} \end{cases} \tag{3.45}$$

Com isto, o PGA pode ser reformulado como:

$$\min \sum_{i=1}^{m} \sum_{k=1}^{k_i} \left(\sum_{j=1}^{n} c_{ij} x_{jk}^i \right) y_k^i \tag{3.46}$$

sujeito a:

$$\sum_{i=1}^{m} \sum_{k=1}^{k_i} x_{jk}^i y_k^i = 1 \quad \forall j \in N = \{1, \cdots, n\} \tag{3.47}$$

$$\sum_{k=1}^{k_i} y_k^i \leq 1 \quad \forall i \in M \tag{3.48}$$

$$y_k^i \in \{0, 1\}, \; i \in M; \; k = 1, ..., k_i \tag{3.49}$$

Esta formulação (3.46)-(3.49) define o problema-mestre a ser considerado. Note que a restrição (3.48) garante que a tarefa j será atribuída a apenas uma máquina.

Nesta formulação, para determinar as atribuições x^i_k possíveis a cada máquina i é necessário resolver os seguintes m problemas da mochila:

$$\min \sum_{j=1}^{n} c_{ij} x_{ij} \tag{3.50}$$

sujeito a:

$$\sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i \tag{3.51}$$

$$x_{ij} \in \{0,1\}, j \in N \tag{3.52}$$

Aplicando a relaxação lagrangeana ao problema (1.10)-(1.13) obtém-se o problema:

$$\min_{x \in S} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} - \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{m} x_{ij} - 1 \right) \tag{3.53}$$

onde $S = \{ x^j = (x_{i1}, ..., x_{in}) / x_{ij} \in \{0,1\}$ e $\sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i , i \in M \}$, que pode ser resolvido separadamente por m problemas do tipo:

$$\min_{x \in S_i} \quad \sum_{j=1}^{n} \left[c_{ij} x_{ij} - \lambda_j (x_{ij} - 1) \right] \tag{3.54}$$

onde $S_i = \{ x^j = (x_{i1}, ..., x_{in}) / x_{ij} \in \{0,1\}$ e $\sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i \}$.

O melhor limitante dado pela relaxação lagrangeana é obtido otimizando-se os multiplicadores lagrangeanos, ou seja, pela seguinte dualização:

$$\max_{\lambda} \quad \min_{x \in S} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{m} x_{ij}\right) \tag{3.55}$$

De forma semelhante, sejam λ_1, λ_2,..., λ_n, os multiplicadores *surrogate* aplicados à restrição (1.11) do problema (1.10)-(1.13). Com isto, a relaxação *surrogate* do problema é dada por:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{3.56}$$

sujeito a:

$$\sum_{j=1}^{n} \lambda_j \sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n} \lambda_j \qquad \text{e (1.12)-(1.13)} \tag{3.57}$$

Seja $t \geq 0$, o valor do multiplicador lagrangeano/*surrogate*. Então a relaxação lagrangeana/*surrogate* do problema (3.56)-(3.57) é dada por:

$$\min_{x \in S} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + t\sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{m} x_{ij}\right) \tag{3.58}$$

Fazendo $t = t^*$, onde t^* é o multiplicador que resolve o problema:

$$\max_{t \geq 0} \quad \min_{s \in S} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + t\sum_{j=1}^{n} \lambda_j \left(1 - \sum_{i=1}^{m} x_{ij}\right) \tag{3.59}$$

tem-se um refinamento dos limitantes para o valor da função-objetivo do problema (1.10)-(1.13).

Após resolver a relaxação de Programação Linear do problema-mestre (3.46)-(3.48), obtida substituindo-se a restrição (3.49) por $y_k^i \in [0,1]$, a busca por novas colunas pode ser realizada resolvendo-se o problema de custo reduzido dado por:

$$\max_{i \in M} (z_{M^i} - \mu_i) \tag{3.60}$$

onde μ_i é o custo dual ótimo associado à restrição (3.48) de viabilidade dos agentes no problema-mestre e z_{M^i} é o valor da solução ótima do seguinte problema da mochila:

$$\min \sum_{j=1}^{n} (c_{ij} - t\,\lambda_j) x_j^i \tag{3.61}$$

sujeito a:

$$\sum_{j=1}^{n} r_{ij} x_j^i \leq b_i \tag{3.62}$$

$$x_j^i \in \{0,1\},\ j \in N \tag{3.63}$$

onde λ_j é o custo dual ótimo associado às restrições de particionamento (3.47) da tarefa j no problema-mestre restrito.

Desta forma, pode-se estabelecer o método de Geração de Colunas para o PGA por meio do seguinte algoritmo:

1) Determine um conjunto inicial de colunas para o problema (3.46)-(3.49);

2) Resolva a relaxação de Programação Linear do problema (3.46)-(3.48) considerando o conjunto atual de colunas, obtendo as variáveis duais λ_j ($j \in N$), referentes às restrições (3.37) e μ_i ($i \in M$), referentes às restrições (3.48);

3) Utilize as variáveis duais λ_j ($j \in N$) para determinar o multiplicador lagrangeano/*surrogate* t que resolve o problema (3.59);

4) Utilize as variáveis duais λ_j ($j \in N$) e o valor do multiplicador t encontrado no passo anterior e resolva os subproblemas (3.61)-(3.63) para cada $i \in M$. Sejam $x^{\hat{k}} \in S$ as colunas referentes às soluções destes subproblemas;

5) Adicione as colunas $x^{\hat{k}}$ com custos reduzidos negativos ao problema-mestre restrito (3.46)-(3.48).

6) Se, no passo anterior, nenhuma nova coluna foi acrescentada ao problema-mestre restrito, então pare. Caso contrário, volte para o passo 2.

Este método pode ser usado também para resolver, de forma aproximada, o Problema de Atribuição de Antenas Celulares a Comutadores. Para isto, pode-se considerar a seguinte formulação para o PAACC:

$$\min \sum_{i\in M}\sum_{j\in N} c_{ij}x_{ij} + \sum_{j\in N}\sum_{k\in N} h_{jk} - \sum_{j\in N}\sum_{k\in N} h_{jk}x_{ij}x_{ik} \tag{3.64}$$

sujeito a:

$$\sum_{i\in M} x_{ij} = 1 \quad \forall\, j\in N \tag{3.65}$$

$$\sum_{j\in N} r_j x_{ij} \le b_i \quad \forall\, i\in M \tag{3.66}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i\in M,\ j\in N \tag{3.67}$$

onde M = { 1, ..., m } é o conjunto de índices dos comutadores, N = { 1, ..., n } é o conjunto de índices das células, c_{ij} representam os custos de cabeamento entre os comutadores i e as células j ($i \in M$, $j \in N$), h_{jk} representam os custos de *handoff* por unidade de tempo entre as células j e k ($j, k \in N$), r_j representa o volume de chamadas de cada célula j ($j \in N$), b_i representa a capacidade de atendimento de chamadas do comutador i ($i \in M$) e $x_{ij} = 1$ se a célula j está atribuída ao comutador i ($i \in M$, $j \in N$) e $x_{ij} = 0$, caso contrário.

Desta forma, o PAACC tem uma estrutura semelhante à do PGA, exceto pelos custos de *handoff* na função-objetivo. Assim, para o método de Geração de Colunas, considera-se a formulação do problema como um problema de particionamento, ou seja:

$$\text{Min} \sum_{s \in S} c_s x_s \tag{3.68}$$

sujeito a:

$$\sum_{s \in S} a_{js} x_s = 1 \quad \forall\, j \in N \tag{3.69}$$

$$\sum_{s \in S} b_{is} x_s \leq 1 \quad \forall\, i \in M \tag{3.70}$$

$$x_s \in \{0,1\} \quad \forall\, s \in S \tag{3.71}$$

onde S é o conjunto de colunas que satisfazem (3.66)-(3.67), onde cada coluna $s \in S$ é da forma: $(a_{1s}, \cdots, a_{ns}, b_{1s}, \cdots, b_{ms})^t$, tal que $a_{js} \in \{0,1\}$ $(j \in N)$ e $\exists\, k \in M$ tal que $b_{ks} = 1$ e $b_{is} = 0, \forall i \in M, i \neq k$, ou seja, $\{a_{1s}, \cdots, a_{ns}\}$ é um conjunto de índices de células atribuídas a um comutador k, e c_s é o custo da coluna s, dado por:

$$c_s = \sum_{i=1}^{n} (c_{ki} + \sum_{j=1}^{n} h_{ij}) a_{is} - \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} a_{is} a_{js} \tag{3.72}$$

Neste caso, após relaxar a restrição (3.65) no sentido lagrangeano/*surrogate* tem-se os seguintes m subproblemas geradores de colunas:

$$\text{Min} \sum_{i=1}^{n} [(c_{ki} + \sum_{j=1}^{n} h_{ij}) - t\, \pi_i] x_{ki} - \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} x_{ki} x_{kj} \tag{3.73}$$

sujeito a:

$$\sum_{j=1}^{n} r_j x_{kj} \leq b_k \tag{3.74}$$

$$x_{kj} \in \{0,1\} \quad \forall\ j \in N \tag{3.75}$$

onde: π_i são as variáveis duais associadas às restrições (3.69) da relaxação de Programação Linear do problema-mestre restrito (3.68)-(3.70).

Assim, o método de GC para o PAACC pode ser estabelecido com o seguinte algoritmo:

1) Determine um conjunto inicial de colunas para o problema (3.68)-(3.71);

2) Resolva a relaxação de Programação Linear do problema (3.68)-(3.70) considerando o conjunto atual de colunas, obtendo as variáveis duais π_j ($j \in N$), referentes às restrições (3.69) e μ_i ($i \in M$), referentes às restrições (3.70);

3) Estipule um valor apropriado para o multiplicador lagrangeano/*surrogate* t;

4) Utilize as variáveis duais π_j ($j \in N$) e o valor do multiplicador t e resolva os subproblemas (3.73)-(3.75) para cada $k \in M$. Seja S' o conjunto de colunas referentes às soluções destes subproblemas;

5) Para cada coluna $s \in S'$, determine o custo reduzido da coluna por: $cr_s = c_s - \sum_{i=1}^{n} \pi_i a_{is} - \mu_k$, onde c_s é o custo da coluna s, conforme estabelecido em (3.72), e acrescente as colunas s tais que $cr_s < 0$ ao problema-mestre restrito (3.68)-(3.70).

6) Se, no passo anterior, nenhuma nova coluna foi acrescentada ao problema-mestre restrito, então pare. Caso contrário, volte para o passo 2.

3.10 O Método *Branch-and-Price* Proposto

Os métodos de Geração de Colunas apresentados na seção anterior resolvem os problemas de atribuição de forma aproximada, pois, a menos que todas as variáveis

y_k^i sejam inteiras, a solução obtida não é factível para o problema original. Assim, faz-se necessário empregar um algoritmo B&B para determinar soluções factíveis. Isto implica em separar o problema em subproblemas, que se traduz em uma estratégia de ramificação para a árvore de busca.

Se y_k^i é fracionária, como comentou-se anteriormente, uma opção é fazer $y_k^i = 0$ em um ramo e $y_k^i = 1$ em outro. Assim, se x_{jk}^i indica se a tarefa j está atribuída à máquina i em um coluna k do problema-mestre restrito, esta decisão implica que em uma ramificação a tarefa j não será de maneira alguma atribuída à máquina i e na outra ramificação pode ser ou não atribuída à máquina i. Outra opção, também comentada anteriormente, seria fazer $x_{ij} = 1$ em um ramo e $x_{ij} = 0$ no outro, ou seja, considerar apenas as atribuições em que a tarefa j está atribuída à máquina i em um ramo e no outro ramo, todas as demais possibilidades de atribuir a tarefa j.

A regra de ramificação deve ser compatível com o subproblema, de modo a evitar a geração de colunas que já foram retiradas pelas ramificações. Isto significa modificar o subproblema de forma que não sejam geradas colunas inviáveis, quando consideradas as ramificações já feitas na árvore, e ainda assim, manter o subproblema tratável. No método B&P proposto neste trabalho para o PGA adotou-se a estratégia de ramificação sugerida por Pigatti (2003):

- para proibir que uma tarefa j seja alocada ao agente i, todas as colunas associadas ao agente i tais que $x_{jk}^i = 1$ devem ser fixadas em zero, ou seja, $y_k^i = 0$, para todo $k \in K_i$.

- para assegurar que uma tarefa j seja alocada ao agente i, todas as colunas associadas ao agente i tais que $x_{jk}^i = 0$ são fixadas em zero, ou seja, $y_k^i = 0$ para todo $k \in K_i$, e todas as colunas associadas a um agente $i' \neq i$ tais que $x_{jk}^{i'} = 1$, são fixadas em zero, ou seja, $y_k^{i'} = 0$ para todo $k \in K_{i'}$.

Com isto, o método B&P proposto para o PGA pode ser estabelecido por meio do seguinte algoritmo:

Seja lista = { problema-mestre inicial };

Enquanto (lista $\neq \varnothing$), repetir:

Recuperar um problema p da lista; Fazer lista = lista – { p };

Resolver o problema p; Seja z o valor da solução;

Se (z >= melhorSolucaoViavel), então fazer:

Podar a árvore no nó correspondente ao problema p;

Senão:

Determinar as linhas r e s onde deverá haver a ramificação;

Sejam p_E o problema resultante de p quando se considera a ramificação da árvore pela esquerda e p_D o problema resultante de p quando se considera a ramificação da árvore pela direita.

Fazer lista = lista $\cup$ { p_E, p_D };

Fim_Se;

Fim_Enquanto.

No próximo capítulo são apresentados os aspectos de implementação dos algoritmos propostos neste capítulo e os resultados computacionais obtidos para o PGA e para o PAACC.

CAPÍTULO 4

IMPLEMENTAÇÃO DOS ALGORITMOS PROPOSTOS E RESULTADOS COMPUTACIONAIS

Este capítulo descreve a implementação dos algoritmos propostos neste trabalho. São descritas: a implementação do algoritmo de Geração de Colunas e do algoritmo *Branch-and-Price* para resolução do Problema Generalizado de Atribuição, e a implementação do algoritmo de Geração de Colunas para resolução do Problema de Atribuição de Antenas Celulares a Comutadores. O capítulo apresenta também os resultados obtidos nos experimentos computacionais realizados. As tabelas que reúnem as informações dos experimentos computacionais comparam os resultados obtidos a partir dos métodos propostos, que utilizam a relaxação lagrangeana/*surrogate*, com os resultados obtidos a partir da relaxação lagrangeana tradicional.

4.1 O Algoritmo de Geração de Colunas para o PGA

O algoritmo de GC para o PGA foi apresentado na Seção 3.9. Para este algoritmo, cada atribuição factível corresponde a uma coluna do problema-mestre restrito (PMR) e é representada por um vetor binário C de tamanho (n + m) da forma: $(a_1, \cdots, a_n, b_1, \cdots, b_m)^t$, onde os elementos $b_1, \cdots, b_m$ indicam a que agente $i \in M = \{ 1, ..., m \}$ esta coluna se refere, ou seja, para um determinado índice $i \in M$ tem-se que: $b_i = 1$ e para $\forall\, k \in M, k \neq i,\ b_k = 0$, e cada um dos elementos a_j com $j \in N = \{ 1, ..., n \}$ indica se a tarefa j está atribuída ($a_j = 1$) ou não ($a_j = 0$) à máquina i.

O conjunto inicial de colunas para o PMR é constituído por um conjunto de n colunas artificiais e um conjunto de colunas que representam uma solução factível para o problema original. Como sugerido por Pigatti (2003), cada coluna artificial corresponde a um vetor nulo, exceto por um elemento que correspondente à restrição de associação de uma das tarefas, que assume o valor 1, conforme ilustrado na FIGURA 4.1. O custo

de cada coluna artificial é estabelecido como o custo da atribuição (agente, tarefa) mais cara.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \begin{matrix} \leftarrow \text{ tarefa 1} \\ \leftarrow \text{ tarefa 2} \\ \\ \leftarrow \text{ tarefa n} \\ \leftarrow \text{ máquina 1} \\ \leftarrow \text{ máquina 2} \\ \\ \leftarrow \text{máquina m} \end{matrix}$$

FIGURA 4.1 – Colunas artificiais do problema-mestre restrito inicial.

O outro conjunto inicial é composto pelas colunas referentes a uma primeira solução inteira encontrada para o problema (3.46)-(3.49) pelo software CPLEX.

Definido o conjunto inicial de colunas, a relaxação de Programação Linear do problema-mestre restrito é resolvida, sendo obtidos os custos duais finais λ_j ($j \in N$), referentes às restrições (3.37), e μ_i ($i \in M$), referentes às restrições (3.48). Novas colunas para o PMR são obtidas determinando-se um valor conveniente para o multiplicador lagrangeano/*surrogate* t e resolvendo-se os m problemas da mochila estabelecidos em (3.61)-(3.63). As colunas para as quais o custo reduzido é negativo, são selecionadas e incorporadas ao problema-mestre restrito. É interessante notar que para cada valor de t ($t \in \Re$) tem-se um novo conjunto de m problemas da mochila e, conseqüentemente, um novo conjunto de m possíveis colunas candidatas a entrarem no PMR. Assim, com a relaxação lagrangeana/*surrogate* é possível gerar um número maior de colunas candidatas, quando comparado com o método tradicional de GC.

4.2 Resultados Computacionais do Método de GC para o PGA

O algoritmo proposto foi implementado em linguagem de programação C e executado em um microcomputador Pentium III com 1.1 GHz e 512 MB de RAM. Foi utilizada a

versão 7.5 do software CPLEX. Para solução dos problemas da mochila utilizou-se o algoritmo de Horowitz-Sahni (Martello e Toth, 1990).

Nos testes computacionais, considerou-se um número máximo de 5000 colunas para o problema-mestre restrito, tendo sido implementado um procedimento de remoção de colunas para o caso do PMR exceder este número máximo de colunas. O procedimento de remoção de colunas considera como improdutivas, as colunas cujos custos reduzidos são maiores do que o custo reduzido médio, consideradas todas as colunas do problema. Para os testes realizados, não se buscou o melhor valor para o multiplicador lagrangeano/*surrogate.* Em vez disto, a cada iteração do algoritmo, foram considerados os seguintes valores para o parâmetro t: 0.50, 0.60, 0.70, 0.80, 0.85, 0.90, 0.93, 0.95, 0.97 e 1.00. Notar que para cada um destes valores de t, são resolvidos m problemas da mochila, a partir dos quais são obtidas as colunas candidatas a entrarem no PMR.

Foram utilizados os seguintes critérios de parada:

- nenhuma nova coluna acrescentada ao PMR;
- $|L_{inf} - L_{PMR}| < 1$, onde L_{inf} corresponde ao melhor limite inferior da solução obtido a partir do dual lagrangeano/*surrogate* e L_{PMR}, corresponde ao limite de Programação Linear obtido para o problema-mestre restrito.

Nos experimentos computacionais realizados foram consideradas instâncias do problema disponíveis na OR-Library (Beasley, 2004) e algumas instâncias de Yagiura (2003), para as quais as soluções são conhecidas. As instâncias consideradas da OR-Library estão divididas em cinco classes: A, B, C, D e E. Os resultados obtidos para estas instâncias são mostrados nas Tabelas 4.1 a 4.5. As instâncias consideradas de Yagiura estão divididas em três classes: C, D e E, e os resultados obtidos para estas instâncias são mostrados nas Tabelas 4.6 a 4.8. Nestas tabelas são mostrados os resultados obtidos com o algoritmo proposto (que utiliza a relaxação lagrangeana/*surrogate*) e, entre colchetes, os resultados obtidos fixando-se t = 1, que corresponde ao método tradicional de GC utilizando a relaxação lagrangeana. Nestas tabelas:

- **m** é o número de agentes;
- **n** é o número de tarefas;
- **Sol** é o valor da solução inteira ótima do problema;
- **Iter** é o número de iterações do algoritmo para resolver o problema;
- **Cols-a** é o número de colunas aproveitadas;
- **Cols-r** é o número de colunas removidas do problema-mestre restrito pelo procedimento de remoção de colunas;
- **LInf** é o limite inferior para o valor da solução;
- **LCplx** é o limite obtido pelo software CPLEX para o problema-mestre restrito;
- **Desv%** é o desvio percentual de LCpx em relação ao valor da solução inteira ótima, ou seja, $\text{Desv}\% = 100 * \frac{|\text{Sol} - \text{LCpx}|}{\text{Sol}}$;
- **Cpu** é o tempo computacional em segundos, e
- **S/L** é a relação entre o tempo de execução do algoritmo proposto (combinado com a relaxação lagrangeana/*surrogate*) e o tempo de execução do algoritmo quando combinado com a relaxação lagrangeana (t = 1).

TABELA 4.1 – Instâncias da Classe A

m	n	Sol	Iter	Cols-a	Cols-r	LInf	LCpx	Desv%	Cpu	S/L
5	100	1698	383 [897]	3575 [4959]	7767 [2162]	1698,00 [1697,04]	1698,00 [1698, 00]	0,000 [0, 000]	125,86 [228,25]	0,55
10	100	1360	68 [176]	4285 [4178]	2222 [0]	1360,00 [1360,00]	1360,00 [1360,00]	0,000 [0,000]	19,33 [30,21]	0,64
20	100	1158	30 [54]	5253 [3193]	20 [0]	1158,00 [1158,00]	1158,00 [1158,00]	0,000 [0,000]	3,47 [5,02]	0,69
5	200	3235	2827 [4478]	3332 [4726]	54337 [25216]	3235,00 [3234,51]	3235,00 [3235,00]	0,000 [0,000]	7217,47 [10715,07]	0,67
10	200	2623	327 [972]	3732 [4028]	15707 [9116]	2622,96 [2623,00]	2623,00 [2623,00]	0,000 [0,000]	679,59 [1580,23]	0,43
20	200	2339	93 [214]	4418 [3791]	10168 [3956]	2339,00 [2338,25]	2339,00 [2339,00]	0,000 [0,000]	165,35 [307,22]	0,54

TABELA 4.2 – Instâncias da Classe B

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
5	100	1843	176 [496]	4553 [4470]	12613 [4064]	1838,01 [1838,03]	1838,84 [1838,84]	0,226 [0,226]	59,45 [192,69]	0,31
10	100	1407	81 [135]	4397 [4074]	3975 [0]	1407,00 [1407,00]	1407,00 [1407,00]	0,000 [0,000]	18,43 [19,58]	0,94
20	100	1166	27 [44]	4132 [3157]	2132 [0]	1166,00 [1166,00]	1166,00 [1166,00]	0,000 [0,000]	3,95 [5,36]	0,74
5	200	3553	1367 [2314]]	4268 [3409	100480 [35220]	3549,01 [3549,02]	3549,34 [3549,53]	0,103 [0,098]	3524,82 [5549,72]	0,64
10	200	2831	196 [431]	4840 [4991]	26111 [7949]	2825,03 [2825,12]	2825,71 [2825,51]	0,187 [0,194]	398,87 [900,12]	0,44
20	200	2340	95 [156]	4562 [4819]	13904 [2480]	2338,04 [2338,10]	2338,57 [2338,52]	0,061 [0,063]	120,85 [219,68]	0,55

TABELA 4.3 – Instâncias da Classe C

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
5	100	1931	185 [404]	3361 [3354]	12585 [3990]	1929,06 [1929,03]	1929,67 [1929,83]	0,069 [0,061]	61,05 [115,72]	0,53
10	100	1402	53 [103]	4957 [4067]	4246 [0]	1399,58 [1399,43]	1399,89 [1399,86]	0,150 [0,153]	7,75 [12,28]	0,63
20	100	1243	21 [34]	4081 [3094]	2373 [0]	1241,50 [1241,20]	1241,67 [1241,80]	0,107 [0,097]	2,33 [2,64]	0,88
5	200	3456	1456 [2432]	4373 [4517]	107906 [33566]	3454,16 [3454,03]	3454,56 [3454,49]	0,042 [0,044]	3282,77 [6055,82]	0,54
10	200	2806	183 [424]	3280 [4743]	25970 [8021]	2803,05 [2803,05]	2803,95 [2803,95]	0,073 [0,073]	331,14 [952,11]	0,35
20	200	2391	67 [144]	4835 [3771]	13948 [3774]	2390,05 [2390,09]	2390,17 [2390,17]	0,035 [0,035]	96,49 [180,57]	0,53

TABELA 4.4 – Instâncias da Classe D

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
5	100	6353	192 [301]	4784 [3915]	6542 [3916]	6349,08 [6349,07]	6349,93 [6349,92]	0,048 [0,048]	64,80 [78,43]	0,83
10	100	6349	42 [66]	4088 [4097]	2231 [0]	6341,17 [6341,39]	6341,45 [6341,45]	0,119 [0,119]	10,98 [10,86]	1,01
20	100	6196	18 [31]	5442 [3571]	20 [0]	6176,09 [6176,14]	6176,14 [6176,14]	0,321 [0,321]	3,23 [3,74]	0,86
5	200	12743	1228 [1343]	3490 [3719]	49561 [27243]	12740,04 [12740,02]	12740,04 [12740,04]	0,023 [0,023]	4449,99 [3495,80]	1,27
10	200	12433	221 [284]	4627 [3470]	17475 [10152]	12425,03 [12425,05]	12425,61 [12425,73]	0,059 [0,058]	495,88 [584,47]	0,85
20	200	12244	58 [93]	3607 [4160]	12662 [4154]	12229,11 [12229,06]	12229,70 [12229,70]	0,117 [0,117]	122,59 [156,01]	0,79

TABELA 4.5 – Instâncias da Classe E

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
5	100	12681	191 [333]	4619 [4294]	7961 [1921]	12673,05 [12673,05]	12673,05 [12673,10]	0,063 [0,063]	66,88 [79,60]	0,84
10	100	11577	71 [102]	3722 [4239]	5447 [0]	11568,01 [11568,02]	11568,02 [11568, 00]	0,078 [0,078]	12,69 [15,15]	0,84
20	100	8436	25 [39]	3482 [3601]	5374 [0]	8431,27 [8431,10]	8431,52 [8431,53]	0,053 [0,053]	4,76 [5,58]	0,85
5	200	24930	1626 [2531]	3813 [4653]	70077 [31003]	24926,03 [24926,07]	24926,69 [24926,60]	0,013 [0,013]	4086,47 [5703,46]	0,72
10	200	23307	369 [499]	3820 [4380]	25744 [9005]	23302,02 [23302,05]	23302,05 [23302,10]	0,021 [0,021]	646,15 [772,70]	0,84
20	200	22379	93 [150]	3673 [3776]	20471 [4823]	22376,03 [22376,16]	22376,79 [22376,80]	0,010 [0,010]	160,75 [219,32]	0,73

TABELA 4.6 – Instâncias da Classe C / Yagiura

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
20	400	4781	408 [602]	4629 [3969]	90997 [24671]	4780,00 [4780,01]	4780,18 [4780,19]	0,017 [0,017]	4898,72 [7231,76]	0,68
40	400	4244	104 [162]	4298 [3468]	52408 [11486]	4243,10 [4243,04]	4243,45 [4243,45]	0,013 [0,013]	1506,70 [1994,91]	0,76

TABELA 4.7 – Instâncias da Classe D / Yagiura

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
20	400	24561	268 [345]	5077 [3625]	59420 [26212]	24560,03 [24560,10]	24560,23 [24560,21]	0,003 [0,003]	6730,81 [6906,88]	0,97
40	400	24350	84 [149]	4050 [4769]	59569 [17795]	24349,04 [24349,42]	24349,50 [24349,50]	0,002 [0,002]	1871,64 [2628,16]	0,71

TABELA 4.8 – Instâncias da Classe E / Yagiura

m	n	Sol	Iter	Colsa	Colsr	LInf	LCpx	Desv%	Cpu	S/L
20	400	44876	489 [647]	3921 [3572]	99763 [29223]	44875,03 [44875,13]	44875,48 [44875,49]	0,001 [0,001]	9075,33 [7466,17]	1,22
40	400	44557	149 [235]	3945 [4948]	82365 [17015]	44556,01 [44556,38]	44556,88 [44556,88]	0,000 [0,000]	2722,58 [3358,14]	0,81

Para as instâncias das classes A e B da OR-Library (Tabelas 4.1 e 4.2) pode-se observar que o tempo computacional reduz-se quase à metade quando é utilizada a relaxação lagrangeana/*surrogate* $(S/L_{médio} = 0{,}595)$. Para as instâncias das classes C, D e E da OR-Library (Tabelas 4.3, 4.4 e 4.5) e de Yagiura (Tabelas 4.6, 4.7 e 4.8), a redução se

deu em proporções menores, mas ainda significativas: $(S/L_{médio} = 0{,}772)$, para as instâncias da OR-Library e $(S/L_{médio} = 0{,}858)$, para as instâncias de Yagiura. Deve-se observar que, mesmo não se buscando o melhor valor para o multiplicador lagrangeano/*surrogate* e considerando 10 valores pré-estabelecidos para o parâmetro t, o que leva à geração de um número bem maior de colunas, a relaxação lagrangeana/*surrogate* supera em velocidade a relaxação lagrangeana tradicional no método de GC. Isto pode ser constatado observando-se a relação entre os valores das colunas Cols-r nas tabelas acima: para as classes A e B, em média são removidas 20786 colunas para a relaxação lagrangeana/*surrogate* e 7514 colunas, para a relaxação lagrangeana; para as classes C, D e E da OR-Library esta relação é de 21700 colunas para a relaxação lagrangeana/*surrogate* contra 7865 colunas, para a relaxação lagrangeana. Para as instâncias de Yagiura esta relação é de 74087 / 21067, para as relaxações lagrangeana/*surrogate* e lagrangeana, respectivamente.

Deve-se observar que este ganho em velocidade não se dá às custas da qualidade da solução obtida, medida pelo valor de Desv%. Pode-se notar que o desvio percentual do limite produzido através da resolução do problema-mestre restrito relaxado linearmente em relação à solução inteira ótima do problema permanece, em média, praticamente o mesmo, para as relaxações lagrangeana/*surrogate* e lagrangeana. Os valores médios de Desv% para as instâncias consideradas nos testes computacionais são mostrados na TABELA 4.9.

TABELA 4.9 – Qualidade das Soluções Obtidas

Classes de Instâncias	Desv% Médio
A, B	0,048 [0,048]
C, D, E / OR-Library	0,078 [0,077]
C, D, E / Yagiura	0,006 [0,006]

4.3 O Algoritmo de Geração de Colunas para o PAACC

No algoritmo de GC para o PAACC considera-se a formulação dada por (3.68)-(3.71). Para gerar as colunas iniciais do problema-mestre restrito resolve-se o problema sem

considerar os custos de *handoff*, ou seja, determina-se a primeira solução inteira obtida pelo software CPLEX para o problema:

$$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \tag{4.1}$$

sujeito a:

$$\sum_{i \in M} x_{ij} = 1 \quad \forall\, j \in N \tag{4.2}$$

$$\sum_{j \in N} r_j x_{ij} \leq b_i \quad \forall\, i \in M \tag{4.3}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i \in M,\, j \in N \tag{4.4}$$

onde M = { 1, ..., m } é o conjunto de índices dos comutadores, N = { 1, ..., n } é o conjunto de índices das células, c_{ij} representa o custo de cabeamento entre o comutador i e a célula j ($i \in M$, $j \in N$), r_j representa o volume de chamadas da célula j ($j \in N$), b_i representa a capacidade de atendimento de chamadas do comutador i ($i \in M$) e $x_{ij} = 1$ se a célula j está atribuída ao comutador i ($i \in M$, $j \in N$) e $x_{ij} = 0$, caso contrário. A solução deste problema, denominada de solução de referência, é uma atribuição viável para o problema original (3.68)-(3.71).

Outras colunas são geradas aleatoriamente até que um número pré-determinado INICOLS de colunas sejam incluídas no problema-mestre inicial. Para isto utiliza-se o seguinte algoritmo:

```
Enquanto (ncols < INICOLS)
    Para i = 1 até m, faça:
        Para j = 1 até n, faça:
            Atribuir a célula j ao comutador i;
            L = { j };
```

cap = $b_i - r_j$;

Gerar aleatoriamente um índice de célula k ($k \notin L$);

Enquanto ($r_k \leq cap$)

Atribuir a célula k ao comutador i;

$L = L \cup \{ k \}$;

cap = cap – r_k ;

Gerar aleatoriamente um novo índice de célula k, ($k \notin L$);

Fim_Enquanto

Fim_Para (j)

Fim_Para (i)

Fim_Enquanto

Uma coluna s corresponde às atribuições de tarefas a um comutador i, representada como um vetor, conforme estabelecido em (3.68)-(3.71). O subproblema gerador de colunas é o problema dado em (3.73)-(3.75), onde π_i são as variáveis duais associadas às restrições (3.69). A dificuldade neste caso está no fato dos m subproblemas serem problemas da mochila com função-objetivo não linear. No entanto, para um determinado k, denominando $z_{ijk} = x_{ki} x_{kj}$ e levando em conta as equações de linearização consideradas por Merchant e Sengupta (1995), pode-se reescrever cada um dos m subproblemas (3.73)-(3.75) como:

$$\text{Min} \sum_{i=1}^{n} [(c_{ki} + \sum_{j=1}^{n} h_{ij}) - t\pi_i] x_{ki} - \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} z_{ijk} \qquad (4.5)$$

sujeito a:

$$z_{ijk} - x_{ki} \leq 0 \qquad (4.6)$$

$$z_{ijk} - x_{kj} \leq 0 \qquad (4.7)$$

$$z_{ijk} - x_{ki} - x_{kj} \geq -1 \tag{4.8}$$

$$\sum_{i=1}^{n} r_i x_{ki} \leq b_k \tag{4.9}$$

$$x_{ki} \in \{0,1\} \quad \forall i \in N \tag{4.10}$$

As seguintes heurísticas foram consideradas para a solução dos subproblemas a cada iteração do algoritmo de GC:

- Heurística **Sub**: que consiste no procedimento que resolve separadamente os m subproblemas dados por (4.5)-(4.10) usando o *software* CPLEX. Deve-se observar que, como os subproblemas são resolvidos separadamente, ao final pode haver células atribuídas a mais de um comutador e mesmo células não atribuídas a comutador algum.

- Heurística **LSH**: uma adaptação do procedimento *Local Search Heuristic* proposto por Menon e Gupta (2004), que consiste em resolver, separadamente, cada subproblema k por meio do seguinte algoritmo:

    1) Ordenar as células em ordem crescente dos valores $(c_{ki} - t\,\pi_i)$, onde t é o multiplicador lagrangeano/*surrogate*;

    2) Alocar as células ao comutador k, segundo a ordenação, enquanto isso for possível, considerando a capacidade b_k do comutador;

    3) Para as células não atribuídas, tentar a atribuição de menor custo ainda possível, respeitando as restrições de capacidade dos comutadores.

- Heurística **GSH**: uma adaptação do procedimento *Global Search Heuristic* proposto por Menon e Gupta (2004), que consiste em resolver todos os m subproblemas por meio do seguinte algoritmo:

1) Ordenar as células em ordem crescente segundo os valores de $(c_{ki} - t\,\pi_i)$, onde t é o multiplicador lagrangeano/*surrogate*;

2) Seja $\{o_1, ..., o_{n*m}\}$ a ordenação dos pares (i, k) de índices das n células atribuídas aos m comutadores;

3) Para j = 1, ..., n*m, faça:

   a. Seja o_j = (i, k);

   b. Alocar a célula i ao comutador k, caso a célula i ainda não esteja atribuída à comutador algum e a capacidade do comutador k não seja violada;

4) Para as células não atribuídas, tentar a atribuição de menor custo ainda possível, respeitando as restrições de capacidade dos comutadores.

– Heurística **Ref**: usada para encontrar soluções viáveis melhores para o problema original, com base nas atribuições obtidas pela heurística Sub e pela solução de referência. A heurística Ref consiste nos seguintes passos:

1) Selecionar dentre as atribuições obtidas através da heurística Sub, aquelas em que uma célula está atribuída a um único comutador;

2) Para cada célula i ainda não atribuída, atribuí-la ao comutador k ao qual esta célula está atribuída na solução de referência, caso isto seja possível, ou seja, se a restrição de capacidade do comutador k não for violada. Caso contrário, procurar atribuir a célula i a um comutador com capacidade para recebê-la ao menor custo possível;

3) Realizar trocas entre as n células e os m comutadores considerando o melhor ganho para a função-objetivo do problema original.

4) Substituir a solução de referência pela solução obtida, caso esta solução seja viável e melhor do que a solução de referência atual.

Após resolver os subproblemas, as colunas candidatas a entrar no problema-mestre restrito são obtidas calculando-se o custo reduzido de cada coluna s da solução dos m subproblemas, ou seja, calculando-se:

$$cr_s = c_s - \sum_{i=1}^{n} \pi_i a_{is} - \mu_k \qquad (4.11)$$

onde c_s é o custo da coluna s dado por (3.72). Se $cr_s < 0$, então a coluna s é adicionada ao problema-mestre restrito. No caso da heurística Ref, se as colunas obtidas proporcionarem uma melhora na solução de referência, então todas as colunas são adicionadas ao problema-mestre restrito, independentemente de seus custos reduzidos.

O método de GC proposto para o PAACC foi implementado utilizando três estratégias de resolução para os subproblemas:

- Estratégia **LG**: a cada iteração, os m subproblemas são resolvidos utilizando-se a heurística LSH. Caso o valor da solução obtida pelo CPLEX para o problema-mestre restrito não se altere, utiliza-se também a heurística GSH;

- Estratégia **GSR**: a cada iteração, os m subproblemas são resolvidos utilizando-se as heurísticas GSH, Sub e Ref;

- Estratégia **GLR**: a cada iteração, os m subproblemas são resolvidos utilizando-se as heurísticas GSH, LSH e Ref.

Em cada uma destas estratégias, todas as colunas obtidas pelas heurísticas são aproveitadas, segundo o critério de custo reduzido negativo estabelecido em (4.11).

Os algoritmos de GC resultantes destas estratégias adotam os seguintes critérios de parada:

- A solução do problema-mestre restrito é menor ou igual à solução ótima do problema obtida pelo CPLEX para a formulação (2.46)-(2.51), substituindo-se a restrição (2.49) pelas restrições de linearização (2.52)-(2.55);

- A solução do problema-mestre restrito não se altera por um número determinado de iterações (MAXREPEAT). Nos testes computacionais realizados considerou-se MAXREPEAT = 10;

- Nenhuma nova coluna é acrescentada ao problema-mestre restrito.

4.4 Resultados Computacionais do Método de GC para o PAACC

O algoritmo de GC proposto para o PAACC foi implementado na linguagem C e executado em um microcomputador Pentium III com 1.1 GHz e 512 MB de RAM. Foi utilizada a versão 7.5 do software CPLEX.

Para o experimento computacional foram consideradas as mesmas instâncias testadas por Menon e Gupta (2004). Para estas instâncias, o número de comutadores varia de 2 a 5 e os problemas estão classificados segundo o número de células nas seguintes em:

- Pequenos: o número de células varia de 15 a 60;
- Médios: o número de células varia de 75 a 150; e
- Grandes: o número de células varia de 175 a 250.

Nestes testes realizados não se buscou o melhor multiplicador lagrangeano/*surrogate.* O valor do parâmetro t foi estabelecido inicialmente em 0.50 e, a cada iteração, este valor foi incrementado de 0.01 até atingir o valor 1.00, quando então é mantido inalterado.

Os resultados obtidos, para cada uma das estratégias LG, GSR e GLR, estão disponíveis nas Tabelas 4.10 a 4.18. Nestas tabelas:

- **n** é o número de células;
- **m** é o número de comutadores;
- **Soluc** é o valor da solução ótima do problema calculada pelo CPLEX;

- **Tempo** é o tempo gasto pelo CPLEX para determinar Soluc;
- **Iter** é o número de iterações do algoritmo de GC;
- **FCols** é o número final de colunas do problema-mestre restrito;
- **LCpx** é o limite de relaxação linear obtido para o problema-mestre restrito;
- **GapC** é o desvio percentual do limite de relaxação linear obtido em relação à solução ótima do problema, ou seja, GapC = 100*(Soluc – LCpx)/Soluc;
- **Viável** é o valor da melhor solução viável encontrada;
- **GapV** é o desvio percentual da melhor solução viável encontrada em relação à solução ótima do problema, ou seja, GapV = 100*(Soluc – Viável)/Soluc;
- **Cpu** é o tempo gasto no método de GC, em segundos.

Nestas tabelas são mostrados os resultados obtidos com o algoritmo proposto (que utiliza a relaxação lagrangeana/*surrogate*), para cada uma das estratégias de resolução dos subproblemas geradores de colunas e, entre colchetes, os resultados obtidos fixando-se t = 1, que corresponde ao método tradicional de GC utilizando a relaxação lagrangeana.

Nas tabelas a seguir, o símbolo (*) indica que o software CPLEX não foi capaz de determinar a solução ótima do problema devido a limitações de memória. Conseqüentemente, para estes problemas, o símbolo também indica que os desvios percentuais não puderam ser calculados.

TABELA 4.10 – Problemas Pequenos (Estratégia LG)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
15	2	130,911	0,25	1 [1]	5 [5]	130,911 [130,911]	0,000 [0,000]	130,911 [130,911]	0,000 [0,000]	0,00 [0,00]
	3	124,624	0,95	1 [1]	6 [7]	124,624 [124,624]	0,000 [0,000]	124,624 [124,624]	0,000 [0,000]	0,00 [0,00]
	4	100,833	1,60	11 [11]	48 [56]	101,523 [101,523]	-0,684 [-0,684]	101,523 [101,523]	-0,684 [-0,684]	0,01 [0,01]
	5	103,895	6,69	1 [1]	13 [13]	103,895 [103,895]	0,000 [0,000]	103,895 [103,895]	0,000 [0,000]	0,00 [0,00]
30	2	328,281	3,42	18 [14]	35 [28]	328,766 [329,642]	-0,148 [-0,415]	328,766 [329,642]	-0,148 [-0,415]	0,02 [0,00]
	3	345,539	0,90	12 [12]	57 [54]	346,228 [346,228]	-0,199 [-0,199]	346,228 [346,228]	-0,199 [-0,199]	0,01 [0,01]
	4	285,995	21,33	11 [11]	64 [64]	287,819 [287,819]	-0,638 [-0,638]	287,819 [287,819]	-0,638 [-0,638]	0,01 [0,01]
	5	245,727	23,56	11 [11]	73 [76]	247,463 [247,463]	-0,706 [-0,706]	247,463 [247,463]	-0,706 [-0,706]	0,00 [0,02]
45	2	940,623	3,02	11 [11]	35 [35]	944,538 [944,538]	-0,416 [-0,416]	944,537 [944,537]	-0,416 [-0,416]	0,00 [0,02]
	3	546,088	10,56	11 [11]	48 [48]	547,907 [547,907]	-0,333 [-0,333	547,907 [547,907]	-0,333 [-0,333]	0,02 [0,01]
	4	507,574	16,18	1 [1]	9 [9]	507,574 [507,547]	0,000 [0,005]	507,574 [507,574]	0,000 [0,000]	0,00 [0,00]
	5	427,953	54,92	11 [11]	77 [82]	431,461 [431,461]	-0,820 [-0,820]	431,461 [431,461]	-0,820 [-0,820]	0,01 [0,01]
60	2	921,914	1,82	5 [4]	13 [13]	921,914 [921,914]	0,000 [0,000]	921,914 [921,914]	0,000 [0,000]	0,00 [0,01]
	3	707,702	64,73	1 [1]	8 [8]	707,702 [707,702]	0,000 [0,000]	707,702 [707,702]	0,000 [0,000]	0,00 [0,00]
	4	729,071	130,96	11 [11]	66 [67]	737,123 [737,123]	-1,104 [-1,104]	737,122 [737,122]	-1,104 [-1,104]	0,03 [0,03]
	5	694,712	87,71	11 [20]	86 [146]	696,933 [696,579]	-0,320 [-0,269]	696,933 [696,579]	-0,320 [-0,269]	0,03 [0,09]
Média			**26,79**	**8 [8]**	**40 [44]**		**-0,336 [-0,349]**		**-0,336 [-0,349]**	**0,01 [0,01]**

TABELA 4.11 – Problemas Médios (Estratégia LG)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
75	2	1537,676	3,79	28 [20]	74 [54]	1537,972 [1537,972]	-0,019 [-0,019]	1537,973 [1537,973]	-0,019 [-0,019]	0,06 [0,04]
	3	995,417	6,05	11 [17]	49 [72]	996,463 [996,217]	-0,105 [-0,080]	996,465 [996,219]	-0,105 [-0,081]	0,03 [0,04]
	4	1392,520	160,30	11 [11]	60 [62]	1401,868 [1401,868]	-0,671 [-0,671]	1401,867 [1401,867]	-0,671 [-0,671]	0,02 [0,02]
	5	955,439	194,81	11 [11]	88 [77]	963,740 [963,740]	-0,869 [-0,869]	963,740 [963,740]	-0,869 [-0,869]	0,04 [0,03]
100	2	1736,039	6,26	1 [1]	5 [5]	1736,038 [1736,038]	0,000 [0,000]	1736,038 [1736,038]	0,000 [0,000]	0,00 [0,00]
	3	1383,437	21,27	3 [3]	19 [19]	1383,437 [1383,437]	0,000 [0,000]	1383,437 [1383,437]	0,000 [0,000]	0,01 [0,01]
	4	1439,013	37,29	11 [11]	66 [63]	1443,100 [1443,100]	-0,284 [-0,284]	1443,101 [1443,101]	-0,284 [-0,284]	0,05 [0,04]
	5	1423,345	626,16	11 [11]	81 [85]	1429,930 [1429,930]	-0,463 [-0,463]	1429,930 [1429,930]	-0,463 [-0,463]	0,06 [0,06]
125	2	2914,480	10,88	2 [2]	7 [7]	2914,480 [2914,480]	0,000 [0,000]	2914,479 [2914,479]	0,000 [0,000]	0,01 [0,00]
	3	2158,554	69,93	11 [11]	47 [47]	2159,022 [2159,022]	-0,022 [-0,022]	2159,024 [2159,024]	-0,022 [-0,022]	0,05 [0,05]
	4	2942,379	92,59	11 [11]	65 [66]	2948,561 [2948,561]	-0,210 [-0,210]	2948,563 [2948,563]	-0,210 [-0,210]	0,07 [0,07]
	5	1942,076	201,56	11 [11]	78 [84]	1947,032 [1947,032]	-0,255 [-0,255]	1947,034 [1947,034]	-0,255 [-0,255]	0,08 [0,08]
150	2	4007,003	24,96	20 [11]	60 [35]	4008,008 [4008,157]	-0,025 [-0,029]	4008,010 [4008,159]	-0,025 [-0,029]	0,14 [0,06]
	3	3226,089	202,19	11 [11]	50 [46]	3226,146 [3226,146]	-0,002 [-0,002]	3226,146 [3226,146]	-0,002 [-0,002]	0,08 [0,08]
	4	2725,686	281,03	11 [11]	64 [65]	2735,005 [2735,005]	-0,342 [-0,342]	2735,007 [2735,007]	-0,342 [-0,342]	0,09 [0,09]
	5	3114,668	1310,38	11 [11]	83 [83]	3118,016 [3118,016]	-0,107 [-0,107]	3118,017 [3118,017]	-0,108 [-0,108]	0,11 [0,11]
Média			**203,09**	**11 [10]**	**56 [54]**		**-0,211 [-0,210]**		**-0,211 [-0,210]**	**0,06 [0,05]**

TABELA 4.12 – Problemas Grandes (Estratégia LG)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
175	2	5122,587	24,64	3 [3]	10 [10]	5122,586 [5122,586]	0,000 [0,000]	5122,581 [5122,581]	0,000 [0,000]	0,02 [0,02]
	3	4797,655	325,96	11 [11]	47 [47]	4817,158 [4817,158]	-0,407 [-0,407]	4817,157 [4817,157]	-0,406 [-0,406]	0,10 [0,10]
	4	3177,499	113,06	11 [11]	64 [66]	3184,612 [3184,612]	-0,224 [-0,224]	3184,613 [3184,613]	-0,224 [-0,224]	0,09 [0,12]
	5	2491,188	115,69	11 [11]	86 [82]	2496,733 [2496,733]	-0,223 [-0,223]	2496,736 [2496,736]	-0,223 [-0,223]	0,14 [0,14]
200	2	5782,849	47,58	1 [1]	5 [5]	5782,848 [5782,848]	0,000 [0,000]	5782,845 [5782,845]	0,000 [0,000]	0,01 [0,01]
	3	4371,127	159,83	18 [15]	75 [65]	4372,585 [4372,569]	-0,033 [-0,033]	4372,582 [4372,564]	-0,033 [-0,033]	0,23 [0,17]
	4	4393,758	344,11	14 [17]	90 [109]	4398,204 [4396,733]	-0,101 [-0,068]	4398,200 [4397,065]	-0,101 [-0,075]	0,20 [0,27]
	5	3768,432	342,59	22 [27]	152 [176]	3768,467 [3768,485]	-0,001 [-0,001]	3768,468 [3768,517]	-0,001 [-0,002]	0,39 [0,48]
225	2	6541,007	56,39	11 [11]	30 [29]	6553,666 [6553,666]	-0,194 [-0,194]	6553,667 [6553,667]	-0,194 [-0,194]	0,13 [0,11]
	3	4956,891	190,77	19 [15]	85 [70]	4966,277 [4963,207]	-0,189 [-0,127]	4966,285 [4963,213]	-0,190 [-0,128]	0,29 [0,21]
	4	5301,878	546,67	11 [11]	68 [65]	5311,296 [5311,296]	-0,178 [-0,178]	5311,296 [5311,296]	-0,178 [-0,178]	0,18 [0,18]
	5	5280,003	828,93	11 [11]	81 [80]	5298,592 [5298,592]	-0,352 [-0,352]	5298,593 [5298,593]	-0,352 [-0,352]	0,22 [0,22]
250	2	8075,777	101,07	15 [15]	38 [35]	8077,204 [8077,204]	-0,018 [-0,018]	8077,210 [8077,210]	-0,018 [-0,018]	0,21 [0,20]
	3	8613,041	447,09	11 [11]	48 [48]	8617,537 [8617,537]	-0,052 [-0,052]	8617,512 [8617,512]	-0,052 [-0,052]	0,18 [0,16]
	4	5834,561	1269,79	11 [11]	66 [68]	5847,674 [5847,674]	-0,225 [-0,225]	5847,661 [5847,661]	-0,225 [-0,225]	0,21 [0,22]
	5	(*)	(*)	11 [11]	87 [89]	4667,162 [4667,162]	(*) [(*)]	4667,166 [4667,166]	(*) [(*)]	0,26 [0,27]
Média			**327,61**	**12 [12]**	**65 [65]**		**-0,146 [-0,140]**		**-0,146 [-0,141]**	**0,18 [0,18]**

TABELA 4.13 – Problemas Pequenos (Estratégia GSR)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
15	2	130,911	0,25	1 [1]	8 [8]	130,911 [130,911]	0,000 [0,000]	130,911 [130,911]	0,000 [0,000]	0,02 [0,02]
	3	124,624	0,95	1 [1]	8 [8]	124,624 [124,624]	0,000 [0,000]	124,624 [124,624]	0,000 [0,000]	0,04 [0,03]
	4	100,833	1,60	7 [11]	51 [65]	98,217 [98,217]	2,594 [2,594]	100,833 [101,238]	0,000 [-0,402]	0,37 [0,76]
	5	103,895	6,69	1 [1]	15 [16]	103,895 [103,895]	0,000 [0,000]	103,895 [103,895]	0,000 [0,000]	0,06 [0,06]
30	2	328,281	3,42	11 [8]	46 [34]	328,281 [328,281]	0,000 [0,000]	328,281 [328,281]	0,000 [0,000]	0,43 [0,34]
	3	345,539	0,90	2 [2]	17 [17]	345,539 [345,539]	0,000 [0,000]	345,539 [345,539]	0,000 [0,000]	0,11 [0,12]
	4	285,995	21,33	11 [11]	77 [78]	287,819 [287,819]	-0,638 [-0,638]	287,819 [287,819]	-0,638 [-0,638]	0,91 [1,37]
	5	245,727	23,56	12 [12]	99 [103]	246,429 [246,429]	-0,286 [-0,286]	246,429 [246,429]	-0,286 [-0,286]	2,04 [1,76]
45	2	940,623	3,02	11 [11]	46 [46]	944,538 [944,538]	-0,416 [-0,416]	944,537 [944,537]	-0,416 [-0,416]	0,63 [0,63]
	3	546,088	10,56	11 [11]	80 [82]	547,907 [547,907]	-0,333 [-0,333]	547,907 [547,907]	-0,333 [-0,333]	1,04 [1,09]
	4	507,574	16,18	1 [1]	11 [11]	507,574 [507,547]	0,000 [0,005]	507,574 [507,574]	0,000 [0,000]	0,13 [0,13]
	5	427,953	54,92	13 [13]	116 [108]	430,535 [430,535]	-0,603 [-0,603]	430,535 [430,535]	-0,603 [-0,603]	2,98 [2,69]
60	2	921,914	1,82	2 [2]	11 [11]	921,914 [921,914]	0,000 [0,000]	921,914 [921,914]	0,000 [0,000]	0,16 [0,23]
	3	707,702	64,73	1 [1]	10 [10]	707,702 [707,702]	0,000 [0,000]	707,702 [707,702]	0,000 [0,000]	0,13 [0,12]
	4	729,071	130,96	12 [12]	98 [99]	736,970 [736,970]	-1,083 [-1,083]	736,970 [736,970]	-1,083 [-1,083]	2,30 [2,33]
	5	694,712	87,71	7 [7]	77 [77]	694,712 [696,066]	0,000 [-0,195]	694,712 [696,066]	0,000 [-0,195]	2,93 [15,11]
Média			**26,79**	**7 [7]**	**48 [48]**		**-0,048 [-0,060]**		**-0,210 [-0,247]**	**0,89 [1,67]**

TABELA 4.14 – Problemas Médios (Estratégia GSR)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
75	2	1537,676	3,79	4 [4]	21 [21]	1537,675 [1537,675]	0,000 [0,000]	1537,675 [1537,675]	0,000 [0,000]	0,68 [0,47]
	3	995,417	6,05	3 [12]	26 [61]	995,417 [995,663]	0,000 [-0,025]	995,418 [995,664]	0,000 [-0,025]	1,09 [2,99]
	4	1392,520	160,30	13 [13]	88 [89]	1400,581 [1400,581]	-0,579 [-0,579]	1400,581 [1400,581]	-0,579 [-0,579]	4,61 [4,20]
	5	955,439	194,81	12 [12]	110 [114]	962,827 [962,827]	-0,773 [-0,773]	962,828 [962,828]	-0,773 [-0,773]	5,94 [6,06]
100	2	1736,039	6,26	1 [1]	6 [6]	1736,038 [1736,038]	0,000 [0,000]	1736,038 [1736,038]	0,000 [0,000]	0,18 [0,24]
	3	1383,437	21,27	3 [3]	23 [23]	1383,437 [1383,437]	0,000 [0,000]	1383,437 [1383,437]	0,000 [0,000]	1,36 [1,46]
	4	1439,013	37,29	12 [12]	104 [103]	1442,919 [1442,919]	-0,271 [-0,271]	1442,920 [1442,920]	-0,272 [-0,272]	7,51 [7,30]
	5	1423,345	626,16	11 [11]	128 [122]	1429,930 [1429,930]	-0,463 [-0,463]	1429,930 [1429,930]	-0,463 [-0,463]	9,40 [11,28]
125	2	2914,480	10,88	2 [2]	11 [11]	2914,480 [2914,480]	0,000 [0,000]	2914,479 [2914,479]	0,000 [0,000]	0,94 [1,07]
	3	2158,554	69,93	2 [2]	15 [15]	2158,553 [2158,553]	0,000 [0,000]	2158,556 [2158,556]	0,000 [0,000]	1,41 [1,44]
	4	2942,379	92,59	16 [17]	125 [132]	2943,083 [2943,083]	-0,024 [-0,024]	2943,085 [2943,085]	-0,024 [-0,024]	21,51 [23,26]
	5	1942,076	201,56	14 [13]	123 [119]	1944,877 [1944,877]	-0,144 [-0,144]	1944,923 [1944,878]	-0,147 [-0,144]	32,83 [28,90]
150	2	4007,003	24,96	3 [3]	15 [15]	4007,003 [4007,003]	0,000 [0,000]	4007,005 [4007,005]	0,000 [0,000]	2,09 [2,03]
	3	3226,089	202,19	11 [11]	74 [72]	3226,146 [3226,146]	-0,002 [-0,002]	3226,146 [3226,146]	-0,002 [-0,002]	12,99 [13,08]
	4	2725,686	281,03	11 [11]	85 [84]	2735,005 [2735,005]	-0,342 [-0,342]	2735,007 [2735,007]	-0,342 [-0,342]	18,40 [17,82]
	5	3114,668	1310,38	14 [14]	153 [149]	3117,019 [3117,019]	-0,075 [-0,075]	3117,020 [3117,020]	-0,076 [-0,076]	44,30 [46,33]
Média			**203,09**	**8** **[9]**	**69** **[71]**		**-0,167** **[-0,169]**		**-0,167** **[-0,169]**	**10,33** **[10,50]**

TABELA 4.15 – Problemas Grandes (Estratégia GSR)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
175	2	5122,587	24,64	2 [2]	10 [9]	5122,586 [5122,586]	0,000 [0,000]	5122,581 [5122,581]	0,000 [0,000]	1,82 [2,01]
	3	4797,655	325,96	11 [11]	73 [74]	4817,158 [4817,158]	-0,407 [-0,407]	4817,157 [4817,157]	-0,406 [-0,406]	13,79 [14,54]
	4	3177,499	113,06	12 [12]	101 [93]	3184,308 [3184,308]	-0,214 [-0,214]	3184,309 [3184,309]	-0,214 [-0,214]	38,89 [39,13]
	5	2491,188	115,69	13 [13]	105 [111]	2496,527 [2496,527]	-0,214 [-0,214]	2496,530 [2496,530]	-0,214 [-0,214]	82,10 [82,99]
200	2	5782,849	47,58	1 [1]	5 [5]	5782,848 [5782,848]	0,000 [0,000]	5782,845 [5782,845]	0,000 [0,000]	1,36 [1,64]
	3	4371,127	159,83	13 [13]	68 [73]	4371,269 [4371,269]	-0,003 [-0,003]	4371,265 [4371,265]	-0,003 [-0,003]	40,55 [40,21]
	4	4393,758	344,11	19 [19]	155 [159]	4394,182 [4394,182]	-0,010 [-0,010]	4394,180 [4394,180]	-0,010 [-0,010]	104,85 [103,94]
	5	3768,432	342,59	6 [14]	67 [111]	3768,432 [3768,450]	0,000 [0,000]	3768,433 [3768,451]	0,000 [-0,001]	57,17 [132,43]
225	2	6541,007	56,39	11 [11]	35 [35]	6553,666 [6553,666]	-0,194 [-0,194]	6553,667 [6553,667]	-0,194 [-0,194]	14,85 [13,84]
	3	4956,891	190,77	5 [7]	40 [48]	4956,891 [4956,891]	0,000 [0,000]	4956,895 [4956,895]	0,000 [0,000]	21,72 [29,73]
	4	5301,878	546,67	11 [11]	89 [90]	5311,296 [5311,296]	-0,178 [-0,178]	5311,296 [5311,296]	-0,178 [-0,178]	56,83 [55,91]
	5	5280,003	828,93	12 [12]	107 [107]	5298,484 [5298,484]	-0,350 [-0,350]	5298,486 [5298,486]	-0,350 [-0,350]	115,67 [115,29]
250	2	8075,777	101,07	5 [14]	23 [51]	8075,775 [8075,792]	0,000 [0,000]	8075,782 [8075,800]	0,000 [0,000]	11,20 [34,67]
	3	8613,041	447,09	12 [12]	60 [60]	8616,596 [8616,596]	-0,041 [-0,041]	8616,571 [8616,571]	-0,041 [-0,041]	56,36 [56,97]
	4	5834,561	1269,79	12 [12]	97 [94]	5847,340 [5847,340]	-0,219 [-0,219]	5847,328 [5847,328]	-0,219 [-0,219]	85,27 [80,85]
	5	(*)	(*)	12 [12]	110 [109]	4666,176 [4666,176]	(*) [(*)]	4666,179 [4666,179]	(*) [(*)]	205,29 [205,84]
Média			**327,61**	**10 [11]**	**72 [77]**		**-0,122 [-0,122]**		**-0,122 [-0,122]**	**56,73 [63,12]**

TABELA 4.16 – Problemas Pequenos (Estratégia GLR)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
15	2	130,911	0,25	1 [1]	7 [5]	130,911 [130,911]	0,000 [0,000]	130,911 [130,911]	0,000 [0,000]	0,00 [0,00]
	3	124,624	0,95	1 [1]	7 [12]	124,624 [124,624]	0,000 [0,000]	124,624 [149,838]	0,000 [-20,232]	0,00 [0,00]
	4	100,833	1,60	9 [7]	49 [54]	99,414 [100,715]	1,407 [0,117]	101,238 [101,238]	-0,402 [-0,402]	0,01 [0,01]
	5	103,895	6,69	1 [1]	15 [20]	103,895 [103,895]	0,000 [0,000]	103,895 [107,126]	0,000 [-3,110]	0,00 [0,00]
30	2	328,281	3,42	13 [13]	36 [32]	329,642 [329,642]	-0,415 [-0,415]	329,642 [329,642]	-0,415 [-0,415]	0,03 [0,04]
	3	345,539	0,90	3 [3]	22 [22]	345,539 [345,539]	0,000 [0,000]	345,539 [345,539]	0,000 [0,000]	0,02 [0,02]
	4	285,995	21,33	14 [13]	94 [95]	286,077 [286,077]	-0,029 [-0,029]	286,077 [286,077]	-0,029 [-0,029]	0,12 [0,11]
	5	245,727	23,56	13 [11]	112 [108]	246,429 [247,463]	-0,286 [-0,706]	246,429 [250,030]	-0,286 [-1,751]	0,18 [0,14]
45	2	940,623	3,02	11 [11]	46 [56]	944,538 [944,538]	-0,416 [-0,416]	944,537 [944,537]	-0,416 [-0,416]	0,06 [0,08]
	3	546,088	10,56	11 [11]	69 [68]	547,907 [547,907]	-0,333 [-0,333]	547,907 [587,588]	-0,333 [-7,600]	0,13 [0,13]
	4	507,574	16,18	1 [1]	11 [13]	507,574 [507,547]	0,000 [0,005]	507,574 [533,417]	0,000 [-5,091]	0,02 [0,02]
	5	427,953	54,92	19 [11]	140 [108]	430,607 [431,461]	-0,620 [-0,820]	430,607 [465,350]	-0,620 [-8,739]	0,81 [0,46]
60	2	921,914	1,82	7 [4]	27 [16]	921,914 [921,914]	0,000 [0,000]	921,914 [921,914]	0,000 [0,000]	0,11 [0,06]
	3	707,702	64,73	1 [1]	10 [11]	707,702 [707,702]	0,000 [0,000]	707,702 [726,079]	0,000 [-2,597]	0,03 [0,02]
	4	729,071	130,96	12 [30]	100 [224]	736,970 [735,456]	-1,083 [-0,876]	734,385 [735,485]	-0,729 [-0,880]	0,62 [1,84]
	5	694,712	87,71	22 [11]	179 [127]	696,255 [696,933]	-0,222 [-0,320]	696,255 [700,437]	-0,222 [-0,824]	2,84 [1,35]
Média			**26,79**	**9 [8]**	**58 [61]**		**-0,125 [-0,237]**		**-0,216 [-3,255]**	**0,31 [0,27]**

TABELA 4.17 – Problemas Médios (Estratégia GLR)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
75	2	1537,676	3,79	4 [5]	20 [24]	1537,675 [1537,675]	0,000 [0,000]	1537,675 [1537,675]	0,000 [0,000]	0,11 [0,28]
	3	995,417	6,05	3 [5]	23 [40]	995,417 [995,416]	0,000 [0,000]	995,418 [995,417]	0,000 [0,000]	0,36 [0,39]
	4	1392,520	160,30	18 [11]	130 [86]	1400,581 [1401,868]	-0,579 [-0,671]	1400,581 [1434,440]	-0,579 [-3,010]	2,05 [1,07]
	5	955,439	194,81	30 [21]	292 [221]	958,836 [959,135]	-0,356 [-0,387]	958,836 [959,135]	-0,356 [-0,387]	8,33 [5,22]
100	2	1736,039	6,26	1 [1]	6 [6]	1736,038 [1736,038]	0,000 [0,000]	1736,038 [1736,038]	0,000 [0,000]	0,17 [0,07]
	3	1383,437	21,27	3 [3]	23 [23]	1383,437 [1383,437]	0,000 [0,000]	1383,437 [1383,437]	0,000 [0,000]	0,68 [0,95]
	4	1439,013	37,29	14 [11]	123 [88]	1442,919 [1443,100]	-0,271 [-0,284]	1442,920 [1448,912]	-0,272 [-0,688]	4,00 [4,88]
	5	1423,345	626,16	11 [11]	114 [106]	1429,930 [1429,930]	-0,463 [-0,463]	1429,930 [1450,727]	-0,463 [-1,924]	7,33 [7,79]
125	2	2914,480	10,88	2 [2]	9 [8]	2914,480 [2914,480]	0,000 [0,000]	2914,479 [2914,479]	0,000 [0,000]	0,42 [0,24]
	3	2158,554	69,93	3 [11]	20 [73]	2158,553 [2159,022]	0,000 [-0,022]	2158,556 [2226,232]	0,000 [-3,135]	1,15 [5,36]
	4	2942,379	92,59	27 [11]	213 [86]	2943,083 [2948,561]	-0,024 [-0,210]	2943,085 [2957,979]	-0,024 [-0,530]	24,46 [11,23]
	5	1942,076	201,56	29 [11]	253 [123]	1944,877 [1947,032]	-0,144 [-0,255]	1944,878 [1991,157]	-0,144 [-2,527]	49,19 [18,37]
150	2	4007,003	24,96	11 [12]	47 [57]	4007,003 [4007,003]	0,000 [0,000]	4007,003 [4007,003]	0,000 [0,000]	4,10 [3,91]
	3	3226,089	202,19	11 [11]	62 [68]	3226,146 [3226,146]	-0,002 [-0,002]	3226,146 [3353,236]	-0,002 [-3,941]	9,14 [9,36]
	4	2725,686	281,03	11 [11]	90 [101]	2735,005 [2735,005]	-0,342 [-0,342]	2735,007 [2744,278]	-0,342 [-0,682]	14,13 [16,06]
	5	**3114,668**	1310,38	15 [11]	168 [120]	3117,019 [3118,016]	-0,075 [-0,107]	3117,020 [3196,399]	-0,076 [-2,624]	34,53 [24,83]
Média			**203,09**	**12 [9]**	**100 [77]**		**-0,141 [-0,171]**		**-0,141 [-1,216]**	**10,01 [6,88]**

TABELA 4.18 – Problemas Grandes (Estratégia GLR)

n	m	Soluc	Tempo	Iter	FCols	LCpx	GapC	Viavel	GapV	Cpu
175	2	5122,587	24,64	3 [3]	13 [16]	5122,586 [5122,586]	0,000 [0,000]	5122,581 [5122,586]	0,000 [0,000]	1,42 [2,02]
	3	4797,655	325,96	11 [11]	61 [65]	4817,158 [4817,158]	-0,407 [-0,407]	4817,157 [5377,756]	-0,406 [-12,091]	12,58 [11,74]
	4	3177,499	113,06	14 [11]	111 [96]	3184,308 [3184,612]	-0,214 [-0,224]	3184,309 [3256,503]	-0,214 [-2,486]	38,00 [30,53]
	5	2491,188	115,69	11 [26]	109 [241]	2496,733 [2496,527]	-0,223 [-0,214]	2496,736 [2496,530]	-0,223 [-0,214]	56,31 [131,25]
200	2	5782,849	47,58	1 [1]	5 [8]	5782,848 [5782,848]	0,000 [0,000]	5782,845 [5783,101]	0,000 [-0,004]	0,71 [0,70]
	3	4371,127	159,83	14 [15]	75 [84]	4371,269 [4371,936]	-0,003 [-0,019]	4371,265 [4371,932]	-0,003 [-0,018]	30,30 [33,65]
	4	4393,758	344,11	23 [27]	207 [223]	4393,760 [4394,182]	0,000 [-0,010]	4393,754 [4394,180]	0,000 [-0,010]	98,94 [115,80]
	5	3768,432	342,59	7 [10]	75 [97]	3768,432 [3768,432]	0,000 [0,000]	3768,433 [3768,433]	0,000 [0,000]	55,12 [76,53]
225	2	6541,007	56,39	11 [11]	43 [40]	6553,666 [6553,666]	-0,194 [-0,194]	6553,667 [6686,105]	-0,194 [-2,218]	12,44 [12,82]
	3	4956,891	190,77	5 [7]	40 [53]	4956,891 [4956,891]	0,000 [0,000]	4956,895 [4956,895]	0,000 [0,000]	16,41 [21,26]
	4	5301,878	546,67	11 [11]	88 [87]	5311,296 [5311,296]	-0,178 [-0,178]	5311,296 [5554,696]	-0,178 [-4,768]	61,01 [61,73]
	5	5280,003	828,93	11 [11]	93 [107]	5298,592 [5298,592]	-0,352 [-0,352]	5298,593 [5746,169]	-0,352 [-8,829]	101,37 [105,44]
250	2	8075,777	101,07	12 [12]	44 [47]	8075,775 [8075,775]	0,000 [0,000]	8075,782 [8075,782]	0,000 [0,000]	18,66 [19,81]
	3	8613,041	447,09	15 [11]	77 [77]	8616,596 [8617,537]	-0,041 [-0,052]	8616,571 [8652,519]	-0,041 [-0,458]	59,48 [44,06]
	4	5834,561	1269,79	11 [11]	79 [92]	5847,674 [5847,674]	-0,225 [-0,225]	5847,661 [6662,019]	-0,225 [-14,182]	82,43 [64,84]
	5	(*)	(*)	11 [11]	116 [101]	4667,162 [4667,162]	(*) [(*)]	4667,166 [4668,354]	(*) [(*)]	163,70 [170,37]
Média			**327,61**	**11 [12]**	**77 [90]**		**-0,122 [-0,125]**		**-0,122 [-3,019]**	**50,56 [56,41]**

Consideradas as três estratégias de resolução dos subproblemas e os valores médios apresentados nas Tabelas 4.10 a 4.18, pode-se constatar que os algoritmos de Geração de Colunas propostos para o PAACC são bem mais rápidos do que o CPLEX e que a versão do algoritmo que utiliza a relaxação lagrangeana/*surrogate*, no que se refere aos indicadores: número de iterações, número final de colunas do problema-mestre restrito, desvio percentual da solução de relaxação linear em relação à solução ótima do problema e tempo computacional, apresenta resultados semelhantes aos obtidos pela versão do algoritmo que utiliza a relaxação lagrangeana tradicional, ainda que, em geral, ligeiramente melhores, como mostra a TABELA 4.19. No entanto, pode-se perceber que a qualidade (medida pelo indicador GapV) da melhor solução viável encontrada pelo algoritmo que utiliza a relaxação lagrangeana/*surrogate* é bem superior do que a encontrada pela outra versão do algoritmo.

TABELA 4.19 – Valores Médios Gerais de Indicadores

Tempo	Iter	FCols	GapC	GapV	Cpu
185,83	9,78 [9,56]	65,00 [65,22]	-0,16 [-0,18]	-0,19 [-0,97]	14,34 [15,45]

Na comparação entre as três estratégias de resolução dos subproblemas, a estratégia LG é muito superior às outras duas em relação ao tempo computacional, embora a qualidade das soluções relaxada (medida pelo GapC) e viável (medida pelo GapV) seja melhor para a estratégia GSR, como mostra a TABELA 4.20.

TABELA 4.20 – Comparação de Estratégias de Resolução de Subproblemas

Estratégia	Iter	FCols	GapC	GapV	Cpu
LG	10,33 [10,00]	53,67 [54,33]	-0,23 [-0,23]	-0,23 [-0,23]	0,08 [0,08]
GSR	8,33 [9,00]	63,00 [65,33]	-0,11 [-0,12]	-0,17 [-0,18]	22,65 [25,10]
GLR	10,67 [9,67]	78,33 [76,00]	-0,13 [-0,18]	-0,16 [-2,50]	20,29 [21,19]

4.5 O Algoritmo de *Branch-and-Price* para o PGA

Implementou-se também, neste trabalho, um algoritmo para o método B&P descrito na Seção 3.10, visando obter soluções exatas para o PGA. Este algoritmo consiste dos seguintes passos:

msv = valor da primeira solução viável encontrada para o problema (3.46)-(3.49) pelo software CPLEX, conforme estabelecido em (4.1);

lista = { problema-mestre restrito inicial };

Enquanto Lista ? ∅, fazer:

RecuperarProblema(rbusca);

EstabelecerColunasUteis();

z = ResolverProblemaMestre();

EstabelecerLimiteDePoda();

Se (z < msv) então:

ResolverProblemaInteiro();

Fim_Se;

Se (não existe possibilidade de poda) então:

DeterminarRamificação(i, j);

pe = RamificarPelaEsquerda(i, j);

IncluirNaLista(rbusca, pe);

pd = RamificarPelaDireita(i, j);

IncluirNaLista(rbusca, pd);

Fim_Se;

Fim_Enquanto.

O algoritmo utiliza a variável **msv** para armazenar o valor da melhor solução viável encontrada. Inicialmente, esta variável armazena o valor da primeira solução inteira obtida pelo CPLEX para o problema (3.46)-(3.49). As colunas referentes a esta primeira solução inteira compõem (juntamente com as colunas artificiais) o problema-mestre restrito inicial, conforme estabelecido na Seção 4.1. O algoritmo utiliza uma **lista** de problemas ativos. Inicialmente, para a raiz da árvore de busca, essa lista contém apenas o problema-mestre restrito inicial. O algoritmo implementado considera esta lista como uma pilha (uma estrutura de dados do tipo LIFO – *Last In First Out*) ou como uma fila (uma estrutura de dados do tipo FIFO – *First In First Out*), o que permite ao algoritmo comportar-se como uma busca em profundidade (se a lista for uma pilha) ou como uma busca em largura (se a lista for uma fila). O parâmetro **rbusca**, utilizado nos procedimentos RecuperarProblema e IncluirNaLista, define o regime de busca desejado. Nos testes realizados considerou-se a busca em profundidade.

O procedimento **RecuperarProblema** recupera o próximo elemento da lista de problemas ativos, removendo-o da lista.

O procedimento **EstabelecerColunasUteis** verifica se as colunas do problema-mestre restrito se aplicam ou não ao problema a ser resolvido, tendo em vista as fixações de variáveis feitas pelas ramificações da árvore. Inicialmente, ou seja, para o nó raiz da árvore, este procedimento considera como úteis todas as colunas do problema-mestre restrito. Cada coluna do problema-mestre corresponde a uma variável de decisão $y_k^i \in [0, 1]$, onde $i \in M$ é um índice de agente e $k \in K_i$ é um índice do conjunto de possibilidades de atribuição de tarefas ao agente i. O CPLEX estabelece, inicialmente, para cada variável y_k^i, o limite inferior fixado em 0 e o limite superior fixado em 1. Na construção da árvore, os ramos da esquerda correspondem à fixação de uma variável $x_{ij} = 1$ e os ramos da direita correspondem à fixação de uma variável $x_{ij} = 0$, onde i

corresponde a um índice de agente e j um índice de tarefa. Assim, fazer $x_{ij} = 1$ na formulação original, corresponde a estabelecer que a tarefa j deve ser atribuída ao agente i e não pode ser atribuída a um outro agente i' ? i. Isto, na formulação do problema-mestre, corresponde a considerar como úteis apenas as colunas onde a tarefa j aparece atribuída ao agente i e as colunas referentes aos agentes i' ? i para as quais a tarefa j não aparece atribuída. Da mesma forma, no ramo da direita, são consideradas úteis as colunas onde a tarefa j não aparece atribuída ao agente i. Assim, no procedimento EstabelecerColunasUteis, uma coluna que não se aplica ao problema é "excluída" do problema-mestre atribuindo-se ao limite superior da variável y_k^i correspondente a essa coluna o valor zero.

O procedimento **ResolverProblemaMestre** corresponde ao algoritmo de Geração de Colunas. Para o nó raiz da árvore, este algoritmo é como o descrito na Seção 4.1. Para os demais nós da árvore, este procedimento precisa considerar restrições adicionais devidas às ramificações já realizadas. Assim, este procedimento chama, em primeiro lugar, o procedimento EstabelecerColunasUteis, que verifica se as colunas do problema-mestre se aplicam ou não ao problema a ser resolvido. Após resolver o problema-mestre com as colunas úteis, o procedimento ResolverProblemaMestre gera novas colunas resolvendo m problemas da mochila. A cada iteração deste procedimento utiliza-se um valor adequado para o multiplicador lagrangeano/*surrogate* t. O melhor valor de t obtido apenas na primeira iteração do procedimento ResolverProblemaMestre para cada nó da árvore. Para isto, emprega-se o algoritmo de busca proposto por Senne e Lorena (2000), descrito na Seção 3.6. Nas demais iterações, o valor do parâmetro t cresce a um passo conveniente até atingir o valor 1. Nos testes computacionais realizados fixou-se o número de passos (np) necessários para que o valor do parâmetro t atinja o valor 1. Com isto, o valor do passo a ser usado será calculado como: $(1 - t_0)/np$, onde t_0 corresponde ao valor do multiplicador obtido na iteração inicial pelo algoritmo de busca. Por exemplo, considerando $t_0 = 0.6$ e np = 5, o valor do passo a ser usado nas demais iterações será dado por: $(1 – 0.6)/5 = 0.08$. Neste caso, o multiplicador t inicia

com o valor 0.6 e é incrementado em 0.08 a cada iteração, até atingir o valor 1. A partir daí, o multiplicador t é mantido inalterado nas demais iterações.

O procedimento **EstabelecerLimiteDePoda** verifica a possibilidade de poda para o nó atual da árvore. O limite de poda de cada problema é estabelecido pelo o limite inferior (L_{inf}) obtido resolvendo-se o problema dual lagrangeano/*surrogate*. Haverá poda do nó da árvore se $L_{inf} \geq (msv - 1)$.

O procedimento **ResolverProblemaInteiro** considera o problema-mestre final, resultante da aplicação do procedimento ResolverProblemaMestre como um problema inteiro, ou seja, considera $y_k^i \in \{0,1\}$, e utiliza o software CPLEX para determinar uma solução. Por razões práticas, considera-se apenas a primeira solução inteira encontrada pelo CPLEX. Esta solução pode não ser válida para o problema original, pois pode incluir alguma coluna artificial. De qualquer forma, seja a solução inteira obtida pelo CPLEX válida ou não, esta solução obtida é passada para o procedimento **AplicarHeuristicaPrimal** que, por meio de trocas, tenta viabilizar e melhorar o valor da variável **msv**, correspondente à solução viável atual.

O procedimento **DeterminarRamificação** determina os valores dos índices i e j para os quais deve ocorrer a ramificação da árvore de busca, conforme discutido na Seção 3.8. A ramificação ocorre quando existe pelo menos uma variável y_k^i fracionária. Neste caso, o procedimento escolhe a coluna para a qual a variável y_k^i seja o mais próximo de 0.5 e que tenha um custo favorável. Para o algoritmo implementado, escolhe-se a variável y_k^i que tiver o menor valor de $\left(\left(1-\left|y_k^i-0.5\right|\right)*100+c_{ij}\right)$. Uma vez escolhida a coluna, ou seja, o índice do agente i, determina-se o índice da tarefa j para o qual o valor de x_{ij} deverá ser fixado. Optou-se por escolher o índice j da primeira tarefa desta coluna que não viole as fixações de variáveis já estabelecidas anteriormente na árvore. A ramificação da árvore pela esquerda corresponde à fixação $x_{ij} = 1$ e a ramificação pela direita, à fixação $x_{ij} = 0$.

Os procedimentos **RamificarPelaEsquerda** e **RamificarPelaDireita** consideram o par (agente i, tarefa j) determinado pelo procedimento DeterminarRamificação e providenciam as ramificações correspondentes da árvore de busca.

O procedimento **IncluirNaLista** leva em conta o parâmetro rbusca e um problema correspondente a um ramo esquerdo ou direito da árvore e inclui este problema na lista de problemas ativos. A inclusão na lista corresponderá a uma operação empilhamento ou a uma operação de enfileiramento, dependendo de se o parâmetro rbusca corresponder à busca em profundidade ou à busca em largura, respectivamente.

4.6 Resultados Computacionais do Método B&P para o PGA

O algoritmo de B&P proposto para o PGA foi implementado na linguagem C e os resultados computacionais foram obtidos em um computador Pentium 4 com 3 GHz, 512 MB de RAM, utilizando-se a versão 7.5 do software CPLEX.

Implementou-se um procedimento de remoção de colunas que é aplicado quando o problema-mestre restrito excede 40000 colunas. Neste caso, removem-se as 10000 piores colunas (com maiores custos reduzidos). Para os experimentos computacionais foram consideradas as instâncias das classes A, B, C, D e E da OR-Library. Os resultados estão mostrados nas Tabelas 4.21 a 4.25. Nestas tabelas:

- **m** é o número de agentes;
- **n** é o número de tarefas;
- **Sol** é a solução ótima ou melhor conhecida para o problema;
- **SEnc** é a solução encontrada pelo algoritmo proposto;
- **NEnc** é o nó da árvore em que a solução SEnc foi encontrada;
- **TEnc** é o tempo para encontrar a solução SEnc;
- **TamArv** é o total de nós visitados na árvore;

- **TÓtimo** é o tempo total para provar o ótimo, em segundos; e
- **np** é o número de passos fixados para o multiplicador lagrangeano/*surrogate* atingir o valor 1.

Nestas tabelas são mostrados os resultados obtidos com o algoritmo proposto (que utiliza a relaxação lagrangeana/*surrogate*) e, entre colchetes, os resultados obtidos fixando-se t = 1. Nestas tabelas, o símbolo (*) significa que o algoritmo excedeu o limite máximo de tempo de execução, estabelecido em 18000 segundos.

TABELA 4.21 – Instâncias da Classe A

m	n	Sol	SEnc	NEnc	TEnc	TamArv	TÓtimo	np
5	100	1698	1698 [1698]	1 [1]	1,48 [1,72]	1 [1]	1,48 [1,72]	6
10	100	1360	1360 [1360]	1 [1]	0,86 [1,14]	1 [1]	0,86 [1,14]	9
20	100	1158	1158 [1158]	1 [1]	0,41 [0,75]	1 [1]	0,41 [0,75]	10
5	200	3235	3235 [3235]	1 [1]	14,35 [20,02]	1 [1]	14,35 [20,02]	6
10	200	2623	2623 [2623]	1 [1]	7,39 [9,05]	1 [1]	7,39 [9,05]	7
20	200	2339	2339 [2339]	1 [1]	4,70 [5,99]	1 [1]	4,70 [5,99]	8
Médias				**1 [1]**	**4,87 [6,45]**	**1 [1]**	**4,87 [6,45]**	

TABELA 4.22 – Instâncias da Classe B

m	n	Sol	SEnc	NEnc	TEnc	TamArv	TÓtimo	np
5	100	1843	1843 [1843]	87 [106]	105,75 [132,44]	121 [151]	160,44 [186,64]	5
10	100	1407	1407 [1407]	1 [1]	0,72 [0,85]	1 [1]	0,72 [0,85]	5
20	100	1166	1166 [1166]	1 [1]	0,48 [0,70]	1 [1]	0,48 [0,70]	7
5	200	3552	**3554** [3552]	136 [284]	7526,55 [12162,27]	186 [381]	(*) [(*)]	2
10	200	2827	2827 [2827]	41 [32]	192,34 [129,86]	55 [71]	261,44 [249,33]	8
20	200	2339	2339 [2339]	1 [3]	5,11 [6,52]	3 [5]	8,42 [8,80]	2
Médias				**45 [71]**	**1305,16 [2072,11]**	**61 [102]**	**86,30 [89,26]**	

TABELA 4.23 – Instâncias da Classe C

m	n	Sol	SEnc	NEnc	TEnc	TamArv	TÓtimo	np
5	100	1931	1931 [1931]	1 [1]	1,39 [1,62]	25 [23]	32,13 [24,75]	7
10	100	1402	1402 [1402]	27 [47]	10,58 [17,35]	49 [67]	16,19 [22,44]	4
20	100	1243	1243 [1243]	17 [31]	3,14 [4,73]	17 [61]	3,14 [8,78]	4
5	200	3456	3456 [3456]	34 [7]	2836,03 [569,72]	65 [97]	4850,24 [5237,39]	7
10	200	2806	2806 [2806]	227 [738]	791,72 [2641,03]	343 [793]	1241,07 [3116,83]	4
20	200	2391	2391 [2391]	26 [16]	40,76 [28,42]	26 [19]	40,76 [30,53]	9
Médias				**55 [140]**	**613,94 [543,81]**	**88 [177]**	**1030,59 [1406,79]**	

TABELA 4.24 – Instâncias da Classe D

m	n	Sol	SEnc	NEnc	TEnc	TamArv	TÓtimo	np
5	100	6353	6353 [6353]	568 [651]	2783,60 [3341,29]	583 [667]	3142,27 [3407,99]	6
10	100	6347	6347 [6347]	2395 [2707]	1946,71 [2261,44]	2715 [2899]	2282,87 [2441,65]	5
20	100	6185	**6188 [6190]**	7782 [11793]	3186,21 [4867,94]	31600 [33471]	(*) [(*)]	4
Médias				**3582 [5050]**	**2638,84 [3490,22]**	**11633 [12346]**	**2712,57 [2924,82]**	

TABELA 4.25 – Instâncias da Classe E

m	n	Sol	SEnc	NEnc	TEnc	TamArv	TÓtimo	np
5	100	12681	12681 [12681]	32 [90]	58,05 [178,06]	101 [179]	283,95 [501,47]	7
10	100	11577	11577 [11577]	81 [170]	45,97 [87,44]	189 [251]	103,58 [142,26]	6
20	100	8436	8436 [8436]	17 [127]	9,55 [50,58]	83 [131]	29,53 [52,14]	5
5	200	24930	24930 [24930]	72 [139]	6386,13 [9724,08]	133 [145]	12013,82 [13424,88]	4
10	200	23307	23307 [23307]	337 [362]	2324,08 [2652,20]	577 [617]	5051,86 [5219,91]	4
20	200	22379	22379 [22379]	6 [9]	55,55 [111,01]	23 [31]	94,85 [161,11]	3
Médias				**91 [150]**	**1479,89 [2133,90]**	**184 [226]**	**2929,60 [3250,30]**	

Como é possível notar pelos valores médios mostrados nas Tabelas 4.21 a 4.25, o algoritmo que utiliza a relaxação lagrangeana/*surrogate* é sempre superior ao algoritmo que utiliza a relaxação lagrangeana tradicional. Os ganhos médios em relação à abordagem tradicional, consideradas as cinco classes, são mostrados na TABELA 4.26.

TABELA 4.26 – Ganhos Médios em Relação à Abordagem Tradicional

NEnc	TEnc	TamArv	TÓtimo
33,15%	20,73%	22,97%	14,34%

O algoritmo proposto, contudo, apresentou alguns problemas. Para o problema com 5 agentes e 200 tarefas da classe B, por exemplo, ambas as versões do algoritmo não conseguiu provar a otimalidade da solução dentro do limite de tempo estabelecido, e para a versão que utiliza a relaxação lagrangeana/*surrogate*, nem mesmo a solução ótima pôde ser encontrada dentro deste limite de tempo. O mesmo ocorreu para o problema com 20 agentes e 100 tarefas da classe D.

Pode-se notar claramente pelos resultados mostrados nas Tabelas 4.21 a 4.25 que quanto maior o número de tarefas, mais difícil é o problema. Mas, fixado o número de tarefas, a complexidade do problema é inversamente proporcional ao número de agentes. Isto pode ser observado para os problemas das classes A, B, C e E. Os problemas da classe D, no entanto, são muito difíceis e isto se deve ao custo de atribuir tarefas aos agentes ser inversamente proporcional aos recursos requeridos por estas atribuições. Assim, os problemas mais difíceis são as instâncias de 200 tarefas da classe D. Estas instâncias não foram resolvidas pelo algoritmo proposto, pois exigiriam um tempo de execução bem maior do que o limite estabelecido.

Observando os resultados apresentados neste capítulo, pode-se constatar que o uso da relaxação lagrangeana/*surrogate* em métodos de Geração de Colunas e, conseqüentemente, em métodos B&P, para os problemas de atribuição considerados, parece ser interessante pois, de um modo geral, produz colunas mais produtivas e leva

os algoritmos a exigirem menor tempo computacional, quando comparados com os métodos tradicionais.

CAPÍTULO 5

CONSIDERAÇÕES FINAIS

Este trabalho discutiu e propôs métodos de solução de dois problemas difíceis de atribuição: o Problema Generalizado de Atribuição (PGA), um dos mais representativos problemas de Otimização Combinatória e para o qual existem muitos trabalhos publicados na literatura, e o Problema de Atribuição de Antenas Celulares a Comutadores (PAACC), um problema mais recente e ainda relativamente pouco estudado. Tanto para o PGA como para o PAACC, foram propostos métodos de Geração de Colunas para a solução aproximada do problema. O trabalho propôs também um método *Branch-and-Price* para a solução exata do PGA. Os métodos propostos utilizam a relaxação lagrangeana/*surrogate* que, como para outros problemas estudados anteriormente (Narciso, 1998; Lorena e Senne, 2004; Senne *et al*., 2005), demonstrou sua superioridade em relação à relaxação lagrangeana tradicional.

O método de Geração de Colunas proposto para o PGA foi testado para problemas com até 40 agentes e 400 tarefas. Para os problemas das classes A, B, C e D da OR-Library pode-se fazer uma comparação dos resultados obtidos pelo método proposto com os resultados obtidos por Narciso (1998). A TABELA 5.1 apresenta os valores médios para:

- **Método** corresponde ao método utilizado, onde GC é o método de Geração de Colunas proposto, e $(L_{t*}SGAP^{\lambda})_a$, $(L_{t*}SGAP^{\lambda})_b$ e $(L_{t*}SGAP^{\lambda})_c$ são os métodos propostos por Narciso (1998), que utilizam otimização por subgradientes e relaxação lagrangeana/*surrogate*;

- **Iter** corresponde ao número de iterações;

- **Gap** corresponde aos desvios percentuais, que no caso do método proposto é calculado como: $Gap = 100 * \left| \frac{Sol - LCpxl}{Sol} \right|$, onde Sol e LCpx são conforme estabelecidos na Seção 4.2, e no caso dos métodos propostos por Narciso (1998)

é calculado como: $\text{Gap} = 100 * \left| \frac{\text{Sol} - \text{LB}}{\text{Sol}} \right|$, onde LB é o melhor limite inferior obtido para o problema;

- **Cpu** é o tempo computacional, em segundos, que para o método GC foi obtido em um microcomputador Pentium III e para os outros métodos foi obtido em uma estação Sun SPARC 5.

TABELA 5.1 – Comparação de Resultados para o PGA

Método	Iter	Gap	Cpu
GC	391,42	0,073	886,10
$(L_{t*}SGAP^{\lambda})_a$	148,71	0,899	2,60
$(L_{t*}SGAP^{\lambda})_b$	433,71	1,992	115,83
$(L_{t*}SGAP^{\lambda})_c$	541,96	2,643	278,84

Nota-se pelos resultados da TABELA 5.1 que, embora os métodos baseados na otimização por subgradientes serem muito rápidos, a qualidade da solução obtida pelo método proposto é muito superior.

Os métodos de Geração de Colunas propostos para o PAACC foram testados para problemas com até 5 agentes e 250 tarefas. A TABELA 5.2 apresenta valores médios considerados todos os problemas e compara os resultados obtidos pelos métodos propostos neste trabalho com os resultados apresentados por Menon e Gupta (2004). Neste tabela:

- **Método** corresponde ao método utilizado, onde GC/LG, GC/GSR e GC/GLR são os métodos de Geração de Colunas propostos, usando as estratégias LG, GSR e GLR, respectivamente, e MG é o método de Menon e Gupta (2004);

- **Gap** corresponde aos desvios percentuais, que no caso dos métodos propostos é calculado como: $Gap = 100 * \left| \frac{LCpx - Viável}{Viável} \right|$, onde LCpx e Viável são conforme estabelecidos na Seção 4.4, e no caso do método MG é calculado como: $Gap = 100 * \left| \frac{UB - LB}{LB} \right|$, onde UB é o valor da melhor solução viável obtida e LB é o valor obtido pela relaxação linear do problema;

- **Cpu** é o tempo computacional, em segundos.

TABELA 5.2 – Comparação de Resultados para o PAACC

Método	Gap	Cpu
GC/LG	0,23	0,08
GC/GSR	0,17	22,65
GC/GLR	0,16	20,29
MG	0,89	0,26

Embora esta comparação não possa ser considerada perfeita, pois os valores de Gap são calculados de forma diferente e os valores de Cpu foram obtidos em ambientes computacionais distintos (microcomputador Pentium III com 1.1 GHz e 512 MB de RAM e software CPLEX versão 7.5, para os métodos propostos, e microcomputador Pentium 4 2.4 GHz e software LINDO, para o método MG), os valores da TABELA 5.2 apresentam alguma evidência sobre o desempenho dos métodos propostos, demonstrando que a utilização da relaxação lagrangeana/*surrogate* é vantajosa em métodos de GC para o PAACC.

O fato dos subproblemas para o PAACC serem problemas da mochila com função-objetivo não-linear dificulta a obtenção de colunas factíveis para o problema-mestre restrito, tornando necessário o uso de heurísticas. A obtenção de uma boa heurística para resolver estes subproblemas continua sendo um grande desafio.

O método de B&P proposto para o PGA foi testado para problemas com até 20 agentes e 200 tarefas. Para os problemas das classes C, D e E da OR-Library, os resultados obtidos podem ser comparados com os obtidos por Nauss (2003), que utiliza cortes e otimização por subgradientes. A TABELA 5.3 apresenta os valores para a solução encontrada (**SEnc**), o tempo computacional, em segundos, para encontrar esta solução (**TEnc**) e o tempo computacional, em segundos, necessário para confirmar a otimalidade da solução (**TÓtimo**), para o método proposto neste trabalho (**B&P**) e para o método proposto por Nauss (**B&B**). O algoritmo B&P foi executado em um computador Pentium IV 3GHz e o algoritmo B&B foi executado em um computador Pentium II 300MHz.

TABELA 5.3 – Comparação de Resultados de Algoritmos Exatos para o PGA

				B&P			**B&B**		
Classe	**m**	**n**	**SÓtimo**	**SEnc**	**TEnc**	**TÓtimo**	**SEnc**	**TEnc**	**TÓtimo**
C	5	100	1931	1931	1,4	32,1	1931	0,3	7,1
	10	100	1402	1402	10,6	16,2	1402	15,5	33,2
	20	100	1243	1243	3,1	3,1	1243	39,7	69,5
	5	200	3456	3456	2836,0	4850,2	3456	36,4	45,8
	10	200	2806	2806	791,7	1241,1	2806	631,1	1081,0
	20	200	2391	2391	40,8	40,8	2392	926,0	-
D	5	100	6353	6353	2783,6	3142,3	6353	170,7	174,7
	10	100	6347	6347	1946,7	2282,9	6349	1023,7	-
	20	100	6185	6188	3186,2	-	6196	2122,9	-
E	5	100	12681	12681	58,1	284,0	12681	44,8	46,1
	10	100	11577	11577	46,0	103,6	11577	58,3	125,4
	20	100	8436	8436	9,6	29,5	8436	1660,5	1770,1
	5	200	24930	24930	6386,1	12013,8	24930	61,5	69,5
	10	200	23307	23307	2324,1	5051,9	23307	357,6	990,2
	20	200	22379	22379	55,6	94,9	22379	988,2	1126,0
	Médias				**1365,3**	**2084,7**		**542,5**	**461,6**

Os resultados da TABELA 5.3 mostram que o algoritmo proposto por Nauss (2003) consegue obter e confirmar a otimalidade das soluções mais rapidamente do que o algoritmo proposto. No entanto, o algoritmo proposto foi capaz de encontrar a solução ótima para 14 dos 15 problemas considerados na TABELA 5.3, ao passo que o algoritmo B&B obteve a solução ótima apenas para 12 destes problemas, dentro do limite de tempo estabelecido (3600 segundos).

Resultados ainda melhores para os problemas das classes C, D e E da OR-Library são apresentados por Pigatti (2003), também utilizando cortes.

A utilização da relaxação lagrangeana/*surrogate* nos métodos de Geração de Colunas para o PGA e para o PAACC propostos são contribuições inéditas deste trabalho. Como conseqüência dos estudos que resultaram neste trabalho, publicou-se um artigo na Revista Pesquisa Operacional, Edição Especial: 60 anos do Professor Nelson Maculan (Lorena *et al*., 2003) com as contribuições apresentadas no Capítulo 3. Um outro trabalho derivado dos estudos realizados foi apresentado no XXIV Encontro Nacional de Engenharia de Produção (Senne *et al*., 2004), recebendo o prêmio de melhor pôster da área de Pesquisa Operacional.

Este trabalho pode ser complementado com outros estudos. A seguir, sugerem-se alguns tópicos para pesquisas futuras:

- Investigar novas formas de obtenção do conjunto inicial de colunas e da influência da utilização do melhor multiplicador lagrangeano/*surrogate* para a geração de novas colunas em métodos de GC para o PGA e PAACC.

- Explorar a utilização de cortes para aumentar a eficiência dos algoritmos propostos para o PGA.

- Desenvolver um algoritmo eficiente para obter soluções factíveis para os problemas da mochila não linear que se apresentam como subproblemas do método de GC para o PAACC.

- Determinar um bom limitante inferior para o método de GC para o PAACC, de modo a possibilitar o desenvolvimento de um algoritmo B&P para este problema.

- Desenvolvimento de algoritmo *Branch-and-Cut-and-Price* para solução exata do PGA e PAACC.

REFERÊNCIAS BIBLIOGRÁFICAS

Aerts, J.; Korst, J.; Spieksma, F. Approximation of a retrieval problem for parallel disks. **Lecture Notes in Computer Science**, v. 2653, p. 178-188, 2003.

Albareda-Sambola, M.; van der Vlerk, M.H.; Fernández, E. **Exact Solutions to a Class of Stochastic Generalized Assignment Problems**. Research report no. 02A11, University of Groningen, Research Institute of Systems, Organisations and Management, Feb. 2002.

Alfandari, L.; Plateau, A.; Tolla, P. A Two-Phase Path-Relinking Algorithm for the Generalized Assignment Problem. In: **MIC2001 - 4th Metaheuristics International Conference**, Porto, Portugal, July 2001.

Amini, M.M.; Racer, M. A rigorous comparison of alternative solution methods for the generalized assignment problem. **Management Science**, v. 40, p. 868-890, 1994.

Baker, B.M. e Sheasby, J. Accelerating the convergency of subgradient optimisation. **European Journal of Operational Research**, v. 117, p. 136-144, 1999.

Balachandran, V. **An integer generalized transportation model for optimal log assignment in computer networks**. Working Paper 34-72-3. Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, 1976.

Barnhart, C.; Johnson, E.L.; Nemhauser, G.L.; Savelsbergh, M.W.P.; Vance, P.H. Branch-and-price: column generation for solving huge integer programs. **Operations Research**, v. 46, n. 3, p. 316-329, 1998.

Beasley, J.E. **OR-Library**. Disponível em: http://www.brunel.ac.uk/depts/ma/research/jeb/orlib/gapinfo.html. Acesso em 23 de dezembro de 2004.

Bokhari, S.H. **Assignment Problems in Parallel and Distributed Computing**. Kluwer Academic Publisher, Boston, USA, 1987.

Burkard, R.E. Selected topics on assignment problems. **Discrete Applied Mathematics**, v. 123, n. 1-3, p. 257-302, Nov. 2002.

Caprara, A.; Lancia, G.; Ng, S.K. Sorting pemutations by reversals through branch-and-price. **INFORMS Journal on Computing**, v. 13, p. 224-244, 2001.

Cattrysse, D.; Salomon, M.; Van Wassenhove, L. A set partitioning heuristic for the generalized assignment problem. **European Journal of Operational Research**, v. 72 p. 167-174, 1994.

Cattrysse, D.G.; Van Wassenhove, L.N. A survey of algorithms for the generalized assignment problem. **European Journal of Operational Research**, v. 60, p. 260-272, 1992.

Cerny, V. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. **Journal Optimization Theory Applications**, v. 45, n. 1, p. 41-51, 1985.

Chu, P.C.; Beasley, J.E. A genetic algorithm for the generalised assignment problem. **Computers and Operations Research**, v. 24, p. 17-23, 1997.

Dantzig. G.B.; Wolfe, P. Decomposition principle for linear programs. **Operations Research**, v. 8, p. 101-111, 1960.

du Merle, O.; Villeneuve, D.; Desrosiers, J. & Hansen, P. Stabilized column generation. **Discrete Mathematics**, v. 194, p. 229-237, 1999.

Farley, A.A. A note on bounding a class of linear programming problems including cutting stock problems. **Operations Research**, v. 38, p. 992-993, 1990.

Fischetti, M.; Lepschy, C.; Minerva, G.; Romanin-Jacur, G.; Toto, E. Frequency assignment in mobile radio systems using branch-and-cut techniques. **European Journal of Operational Research**, v. 123, p. 241-255, 2000.

Fisher, M.L.; Jaikumar, R.; Van Wassenhove, L.N. A multiplier adjustment method for the generalized assignment problem. **Management Science**, v. 32, n. 9, p. 1095-1103, 1986.

French, A.P.; Wilson, J.M. Heuristic Solution Methods for the Multilevel Generalized Assignment Problem. **Journal of Heuristics**, v. 8, n. 2, p. 143-153, Mar. 2002.

Garey, M.R.; Johnson, D.S. Computers and intractability: A guide to the theory of NP-completeness. San Francisco: W.H. Freeman and Co., 1979. 340p.

Glover, F.; Taillard, E.; de Werra, D. A user's guide to tabu search. **Annals of Operations Research**, v. 41, n. 3, p. 3-28, 1993.

Grundel, D.A.; Krokhmal, P.; Oliveira, C.A.S.; Pardalos, P.M. On the average case behavior of the multidimensional assignment problem **Pacific Journal of Optimization**, v. 1, n. 1, p. 39-57, 2005.

Guignard, M.; Rosenwein, M. An improved dual-based algorithm for the generalized assignment problem. **Operations Research**, v. 37, n. 4, p. 658-663, 1989.

Hung, M.S.; Rom, W.O. Solving the assignment problem by relaxation. **Operations Research**, v. 28, n. 4, p. 969-982, 1980.

Ignácio, A.A.V.; Ferreira Filho, V.J.M.; Galvão, R.D. **Métodos heurísticos num entorno paralelo**. In: Simpósio Brasileiro de Pesquisa Operacional, 32. Anais. Universidade Federal de Viçosa, 2000, p. 769-788.

Kelley, J. The cutting plane method for solving convex programs. **Journal of SIAM**, v. 8, n. 4, p. 703-712, 1960.

Kirkpatrick, S.; Gelatt Jr., C.D.; Kirkpatrick, M.P. Optimization by simulated annealing. **Science**, v. 220, p. 671-680, 1983.

Kirubarajan, T.; Bar-Shalom, Y.; Pattipati, K.R. Multiassignment for tracking a large number of overlapping objects. **IEEE Transactions on Aerospace and Electronic Systems**, v. 37, n. 1, p. 2-21, Jan. 2001.

Koopmans, T.C.; Beckmann, M.J. Assignment problems and the location of economic activities. **Econometrica**, v. 25, p. 53-76, 1957.

Kuhn, H.W. The Hungarian method for the assignment problem. **Naval Research Logistic Quaterly**, v. 2, p. 83-97, 1955.

Laguna, M.; Kelly, J.P.; Gonzalez-Velarde, J.L.; Glover, F. Tabu search for the multilevel generalized assignment problem. **European Journal of Operations Research**, v. 42, p. 677–687, 1995.

Loiola, E.M.; Abreu, N.M.M.; Boaventura Netto, P.O. Uma revisão comentada das abordagens do problema quadrático de alocação. **Pesquisa Operacional**, v. 24, n. 1, p. 73-109, Jan.-Abr. 2004.

Lorena, L.A.N., Pereira, M.A.; Salomão, S.N.A. A relaxação lagrangeana/surrogate e o método de geração de colunas: novos limitantes e novas colunas. **Revista Pesquisa Operacional**, v. 23, n. 1, p. 29-47, 2003. Edição especial: 60 anos – Professor Nelson Maculan.

Lorena. L.A.N.; Senne, E.L.F. A column generation approach to capacitated p-median problems. **Computers and Operations Research**, v. 31, n. 6, p. 863-876, 2004.

Luan, F.; Yao, X. **Solving real-world lecture room assignment problems by genetic algorithms**, Complex Systems - From Local Interactions to Global Phenomena, IOS Press, Amsterdam, p.148-160, 1996.

Martello, S.; Toth, P. An algorithm for the generalized assignment problem. **Operational Research**, v. 81, p. 589-603, 1981.

Martello, S.; Toth, P. **Knapsack problems: algorithms and computer implementations.** New York, Wiley, 1990.

Menon, S.; Gupta, R. Assigning cells to switches in cellular networks by incorporating a pricing mechanism into simulated annealing. **IEEE Transactions on Systems, Man, and Cybernetics, Part B**, v. 34, n. 1, p. 558-565, 2004.

Merchant, A.; Sengupta, B. Assignment of cells to switches in PCS networks. **IEEE/ACM Transactions on Networks**, v. 3, n. 5, p. 521-526, 1995.

Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, A.; Teller, E. Equation of state calculations by fast computing machines. **Journal of Chemical Physics**, v. 21, n. 6, p. 1087-1092, 1953.

Narciso, M.G. **A relaxação lagrangeana/surrogate e algumas aplicações em otimização combinatória**. 1998. 134p Tese (Doutorado em Computação Aplicada) Instituto Nacional de Pesquisas Espaciais, São José dos Campos. 1998.

Narciso, M.G.; Lorena, L.A.N. Lagrangean/surrogate relaxation for generalized assignment problems. **European Journal of Operational Research**, v. 114, p. 165-177, 1999.

Nauss, R.M. Solving the generalized assignment problem: An optimizing and heuristic approach. **INFORMS Journal on Computing**, v. 15, n. 3, p. 249-266, 2003.

Neame, P.J. **Nonsmooth dual methods in integer programming.** Tese de Doutorado, Universidade de Melbourne, Austrália, 1999.

Nemhauser, G.L.; Wolsey, L.A. **Integer and combinatorial optimization**. Wiley-Interscience Series in Discrete Mathematics and Optimization, 1998.

Nowakovski, J.; Schwärzler, W.; Triesch, E. Using the generalized assignment problem in scheduling the ROSAT space telescope. **European Journal of Operational Research**, v. 112, n. 3, p. 531-541, 1999.

Osman, I.H. Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches, **OR Spektrum**, v. 17, p. 211-225, 1995.

Osorio, M.A.; Laguna M. Logic cuts for multilevel generalized assignment problems. **European Journal of Operational Research**, v. 151, n. 1, p. 238-246, Nov. 2003.

Pierre, S.; Houéto, F. Assigning cells to switches in cellular mobile networks using taboo search. **IEEE Transactions on Systems, Man and Cybernetics**, Part B, v. 32, n. 3, p. 209-224, 2002.

Pierskalla, W.P. The multidimensional assignment problem. **Operations Research**, v. 16, p. 422-431, 1968.

Pigatti, A.; Aragão, M.; Uchoa, E. **Stabilized branch-and-cut-and-price for the generalized assignment problem.** Disponível em: http://www.optimization-online.org/db_html/2004/11/990.html. Acesso em 23 de dezembro de 2004.

Pigatti, A.A. **Modelos e algoritmos para o problema de alocação generalizada e aplicações**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Julho 2003.

Pusztaszeri, J.; Rensing, P.E.; Liebling, T.M. Tracking elementary particles near their primary vertex: A combinatorial approach. **Journal of Global Optimization**, v. 9, n. 1, p. 41-64, July 1996.

Quintero, A.; Pierre S. A memetic algorithm for assigning cells to switches in cellular mobile networks. **IEEE Communication Letters**, v. 6, n. 11, p. 484-486, 2002.

Quintero, A.; Pierre S. Assigning cells to switches in cellular mobile networks: a comparative study. **Computer Communications**, v. 26, n. 9, p. 950-960, 2003.

Ross, G.T.; Soland, R.M. A branch-and-bound algorithm for the generalized assignment problem. **Mathematical Programming**, v. 8, p. 91-103, 1975.

Ruland, K.S. A model for aeromedical routing and scheduling. **International Transactions in Operational Research**, v. 6, n. 1, p. 57-73, 1999.

Ryan, D.M.; Foster, B.A. **An integer programming approach to scheduling.** In: Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling. A. Wren (ed.), Amsterdam, 1981, p. 269-280.

Saha, D.; Mukherjee, A.; Bhattacharya, P. A simple heuristic for assignment of cells to switches in a PCS network. **Wirelles Personal Communications**, v. 12, n. 3, p. 209-223, 2000.

Savelsbergh, M. A branch-and-price algorithm for the generalized assignment problem. **Operations Research**, v. 45, n. 6, p. 831-841, 1997.

Senne, E.L.F.; Lorena, L.A.N. **Lagrangean/surrogate heuristics for p-median problems.** In: Computing tools for modeling, optimization and simulation: interfaces in computer science and operations research. M. Laguna and J.L. Gonzalez-Velarde (eds.). Kluwer Academic Publishers, 2000, p. 115-130.

Senne, E.L.F.; Lorena, L.A.N.; Pereira, M.A. A branch-and-price approach to p-median problems. **Computers and Operations Research**, v. 32, n. 6, p. 1655-1664, 2005.

Senne, E.L.F.; Lorena, L.A.N.; Salomão, S.N.A. **Uma abordagem de geração de colunas para o problema generalizado de atribuição**. Anais do XXIV Encontro Nacional de Engenharia de Produção (ENEGEP), 2004.

Vance, P.H.; Barnhart, C.; Johnson, E.L.; Nemhauser, G.L. Solving Binary Cutting Stock Problems by Column Generation and Branch-and-Bound. **Computational Optimization and Applications**, v. 3, p. 111-130, 1994.

Wang, L.; Gu, W. Genetic algorithms with stochastic ranking for optimal channel assignment in mobile communications. **Lecture Notes in Computer Science**, v. 3314, p. 154-159, 2004.

Wilson, J.M. A genetic algorithm for the generalised assignment problem. **Journal of the Operational Research Society**, v. 48, n. 8, p. 804-809, 1997.

Wright, M.B. Speeding up the Hungarian algorithm. **Computers and Operations Research**, v. 17, n. 1, p. 95-96, 1990.

Yagiura, M. **Generalized assignment problem instances**. Disponível em: http://www-or.amp.i.kyoto-u.ac.jp/members/yagiura/gap/. Acesso em 10 de janeiro de 2003.

Yagiura, M.; Ibaraki, T.; Glover, F. **A path relinking approach with ejection chains for the generalized assignment problem** Technical Report 2004-002, Department of Applied Mathematics and Physics, Kyoto University, 2004a.

Yagiura, M.; Ibaraki, T.; Glover, F. An ejection chain approach for the generalized assignment problem. **Informs Journal on Computing**, v. 16, n. 2, p. 133-151, Spring 2004b.

APÊNDICE A

GERAÇÃO DE PROBLEMAS TESTES PARA O PAACC

Este apêndice apresenta como foram gerados os problemas testes para o PAACC, com a finalidade de comparar os algoritmos propostos com os já existentes na literatura. São discutidos a seguir os modelos simuladores de PAACC propostos por Merchant e Sengupta (1995) e por Quintero e Pierre (2003).

A.1 Problemas Testes de Merchant e Sengupta

Na construção dos problemas testes, Merchant e Sengupta (1995) consideram que as células têm formatos hexagonais e de mesmas dimensões. Cada célula tem uma antena em seu centro geométrico e algumas delas têm comutadores alocados em seu centro. A distribuição de comutadores é uniforme e gerada aleatoriamente. Se dois ou mais comutadores estão muito próximos, rejeita-se a atribuição e gera-se uma nova distribuição. O custo de cabeamento é proporcional à distância geométrica entre a antena e o comutador.

A taxa de chamadas originárias de cada célula apresenta distribuição Gamma com média 1 e coeficiente 0,25. Os tempos de duração de uma chamada de dentro de uma célula foram gerados pela distribuição Exponencial com média 1. Se uma célula tem k vizinhos, então divide-se o intervalo (0,1) em k+1 subintervalos selecionando k números aleatórios de uma distribuição uniforme entre 0 e 1. O comprimento do i^{o} intervalo representa a probabilidade de *handoff* de uma célula e seu i^{o} vizinho (i = 1, ..., k).

Considera-se a rede de Jackson para encontrar um conjunto consistente de *handoffs* e volume de chamadas, onde as células são nós servidores em número infinito, a origem das chamadas se comporta tal como a chegada de usuários, a probabilidade de *handoff* é a probabilidade de transição de nós, e o tempo de *holding* das chamadas em cada célula tem média de serviço necessário para um cliente em um nó correspondente.

Foram dados comprimentos das filas, que transladam em um volume de chamada (λ_i) para celulares e a taxa de clientes movendo entre células, que fazem as taxas de *handoff* proporcionais a h_{ij}. Assumiu-se que as capacidades dos comutadores são iguais, baseados na capacidade máxima entre 10 e 50%.

A.2 Problemas Testes de Quintero e Pierre

Quintero e Pierre (2003) também consideram que as células são arranjadas em uma grade hexagonal de comprimentos e larguras quase iguais. As antenas estão localizadas no centro das células e distribuídas igualmente na grade. Contudo, quando duas ou mais antenas estão próximas, o arranjo de antenas é rejeitado e um novo arranjo é escolhido. O custo de cabeamento entre uma célula e um comutador é proporcional à distância que os separa. O coeficiente de proporcionalidade é tomado como sendo igual a uma unidade. A taxa de chamadas λ_i da célula i segue uma distribuição Gamma de média e variância igual a uma unidade. A duração das chamadas dentro das células são distribuídas de acordo com uma distribuição exponencial de parâmetro igual a 1.

Se uma célula j tem k vizinhos, o intervalo [0,1] é dividido em k+1 subintervalos escolhendo k números aleatórios distribuídos igualmente entre 0 e 1. No fim de cada período de serviço na célula j, a chamada seria transferida para o i-ésimo vizinho (i = 1, ..., k) com uma probabilidade de *handoff* igual ao comprimento do i-ésimo intervalo.

Para encontrar o volume de chamadas e a uma taxa coerente de *handoff*, considera-se que um sistema de filas do modelo M/M/1 formando uma rede de Jackson. As taxas de chegadas α_i nas células são obtidas resolvendo o seguinte sistema:

$$\alpha_i - \sum_{j=1}^{n} \alpha_i r_{ij} = \lambda_i \qquad \text{com } i = 1, ..., n. \tag{A.1}$$

Se a taxa de chegadas α_i é maior que a taxa de serviço, a distribuição é rejeitada e escolhida novamente. A taxa de *handoff* h_{ij} é dada por:

$$h_{ij} = \lambda_i r_{ij} \tag{A.2}$$

Todos os comutadores têm a mesma capacidade M, calculada como segue:

$$M = \frac{1}{m}\left(1+\frac{K}{100}\right)\sum_{i=1}^{n}\lambda_i \qquad (A.3)$$

onde K é uniformemente escolhido entre 10 e 50, o que assegura um excesso global de 10 à 50% da capacidade dos comutadores comparado ao volume de chamadas da célula.

A.3 A Distribuição Gamma

A distribuição Gamma aparece em muitas aplicações de simulação. Diz-se que x tem uma distribuição Gamma com parâmetros α e β, denotando-se por χ = Gamma(α, β) com função densidade dada por: $p(x/\alpha,\beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1}e^{-\beta x}$, x > 0, e zero, para x ≤ 0.

Os parâmetros α e β são chamados, respectivamente, de *forma* e *escala* e são ambos positivos (α, β > 0). Além disso, E(x) = α/β e Var(x) = α/β^2.

Quando α é um número inteiro k a Gamma(k, B), onde B = $1/\beta$, é a distribuição da soma de k variáveis exponenciais com parâmetro B. E se U(1), ..., U(k) são variáveis aleatórias uniformes (0,1) independentes,

$$x = -B\sum_{i=1}^{k}\ln(U_i) \text{ tem distribuição Gamma(k, B)} \qquad (A.4)$$

No caso descrito em Merchant e Sengupta (1995), as chamadas são geradas em cada célula através de uma distribuição Gamma de média 1 e coeficiente de variação V = 0.25. Neste caso, tem-se que:

$$E(x) = 1 = \alpha/\beta \qquad (A.5)$$

e

$$V = 0.25 = s / E(x), \qquad (A.6)$$

onde s é o desvio padrão, determinado por:

$$s = \sqrt{var(x)} = \sqrt{\frac{\alpha}{\beta^2}} = \frac{\sqrt{\alpha}}{\beta} \tag{A.7}$$

Tem-se, então, o seguinte sistema de equações:

$$\begin{cases} \frac{\alpha}{\beta} = 1 \\ \frac{\frac{\sqrt{\alpha}}{\beta}}{\frac{\alpha}{\beta}} = 0.25 \end{cases} \tag{A.8}$$

ou seja:

$$\begin{cases} \alpha = \beta \\ \frac{\sqrt{\alpha}}{\beta} = \frac{\alpha}{\beta} 0.25 \end{cases} \tag{A.9}$$

Efetuando os cálculos, tem-se que $\alpha = \beta = 16$, que são os parâmetros de forma e escala, respectivamente. Assim, tem-se que B = 1/16 e x pode ser gerada por:

$$x = -\frac{1}{16} \sum_{i=1}^{k} \ln(U_i) \tag{A.10}$$