Universidade Federal do Ceará

Centro de Ciências

Departamento de Computação

Pós Graduação em Ciência da Computação

Mestrado em Ciência da Computação

Integração de Heurísticas Lagrangeanas com Algoritmos Exatos para a Otimização de Particionamento de Conjuntos

Alexsandro de Oliveira Alves

Agosto de 2007

Universidade Federal do Ceará

Centro de Ciências

Departamento de Computação

Mestrado em Ciência da Computação

Integração de Heurísticas Lagrangeanas com Algoritmos Exatos para a Otimização de Particionamento de Conjuntos

Alexsandro de Oliveira Alves

(alexsandro@lia.ufc.br)

Dissertação submetida ao Programa de Mestrado e Doutorado em Ciência da Computação da Universidade Federal do Ceará como requisito parcial para obtenção de grau de Mestre em Ciência da Computação.

Orientador: *Rafael Castro de Andrade*

Fortaleza, agosto de 2007

Dedico este trabalho a Deus, pela sua infinita graça e misericórdia.

Aos meus pais, pelo apoio moral e financeiro.

E, de forma especial, à minha esposa Fatiélia Frota pelo carinho compartilhado e pelos incansáveis incentivos em todas as etapas do processo de construção deste trabalho.

Agradecimentos

Em primeiro lugar, agradeço a Deus, meu único e suficiente Salvador, pelo dom da vida. Agradeço por ser meu refúgio e minha proteção e por me dar discernimento e serenidade nos momentos decisivos da minha vida.

Agradeço aos meus pais por proporcionarem a minha educação e formação acadêmica. Agradeço aos meus familiares pelos incentivos e por sempre estarem dispostos a me ajudar.

Agradeço à minha esposa Fatiélia Frota, pelo amor, carinho, respeito, compreensão e companheirismo que, juntamente com a benção e graça de Deus, me sustentam e me incentivam a sempre acreditar e lutar pelos meus sonhos e objetivos.

Agradeço a todos os professores da Universidade Federal do Ceará – UFC e da Universidade Estadual do Ceará – UECE que foram responsáveis pela minha formação acadêmica, e, em especial, ao Prof. Marcos José Negreiros Gomes que além de educador e professor, tornou-se um amigo que sempre acreditou na minha capacidade, perseverança e seriedade. Sempre que dele precisamos, ele está disposto a ajudar.

Agradeço ao meu orientador Prof. Rafael Castro de Andrade, por ter aceitado me orientar a partir do segundo ano do Mestrado, após decisões difíceis e desafiadoras que tive que tomar. Agradeço por sua orientação, apoio e compreensão.

Agradeço aos membros da banca por terem aceito prontamente o convite para participar da comissão examinadora desta dissertação de Mestrado.

Agradeço aos meus colegas do grupo de pesquisa ParGO (Paralelismo, Grafos e Otimização) e do LAB2 pela troca de experiências, em especial ao Ítalo, Aline e Pablo com quem tive maior proximidade.

Agradeço a todos os colegas e amigos da UECE, que junto comigo constituíram a maior turma de graduação em Ciência da Computação da UECE.

Agradeço à FUNCAP – Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico pelo financiamento da minha bolsa de estudos durante o Mestrado.

Meus sinceros e humildes agradecimentos a todos vocês.

Resumo

Neste trabalho avaliamos métodos heurísticos e exatos para o Problema de Particionamento de Conjuntos (PPC). Realizamos testes computacionais com heurísticas lagrangeanas baseadas em algoritmos gulosos, busca tabu e método de otimização pelo subgradiente. Os resultados obtidos, comparados com os da literatura, comprovam a eficiência de nossas heurísticas na obtenção de limites inferiores e superiores de boa qualidade, em tempo computacional razoável, para instâncias da literatura. Utilizamos um esquema de *Branch and Bound* para tentar resolver instâncias do PPC à otimalidade e para comprovar a qualidade dos resultados alcançados por nossas heurísticas.

PALAVRAS CHAVE: Particionamento de conjuntos. Busca tabu. Heurísticas lagrangeanas. Método do subgradiente. *Branch and Bound*.

Abstract

In this work we evaluate both exact and heuristic methods for the set partitioning problem (SPP). These heuristics are based on greedy algorithms, tabu search and subgradient optimization. Computational experiments performed on benchmark instances of the problem indicate that our heuristics are competitive with existing ones from the literature in obtaining both lower and upper bounds of good quality in reasonable execution time. We use a Branch and Bound algorithm that allows to prove optimality of solutions obtained by our heuristics for a large set of benchmark instances of the SPP. Thus, we show that our heuristics are efficient in obtaining feasible solutions of good quality for this problem.

KEYWORDS: Set Partitioning. Tabu Search. Lagrangian Heuristics. Subgradient method. Branch and Bound.

Sumário

Lista de Algoritmos

Capítulo 1

Introdução

Neste trabalho apresentamos heurísticas lagrangeanas baseadas em algoritmos construtivos gulosos, meta-heurística busca tabu e otimização pelo subgradiente para obtenção de limites inferiores e superiores para o Problema de Particionamento de Conjuntos (PPC). Um esquema de *Branch and Bound* é empregado na tentativa de provar a otimalidade de soluções encontradas por nossas heurísticas.

O PPC é um problema de otimização combinatória NP-difícil [25] que possui aplicações práticas tais como escalonamento de tripulação de linhas aéreas [32], roteamento de veículos [24], redistribuição política de território [43], localização de facilidades [7], investimento de capitais e projeto de circuitos elétricos [43].

As principais motivações para a realização deste trabalho são desenvolver métodos capazes de obter boas soluções viáveis para o PPC, investigando com maior profundidade o uso de algoritmos meta-heurísticos associados a algoritmos exatos para resolver instâncias da literatura.

A inovação deste trabalho está na forma como as técnicas de resolução (algoritmos gulosos, busca tabu e método de otimização pelo subgradiente) foram organizadas para, em conjunto, obter limites inferiores e superiores de boa qualidade para instâncias da literatura. Os resultados obtidos mostram que nossas heurísticas são competitivas em relação às existentes na literatura. Utilizamos um esquema de *Branch and Bound* para comprovar a qualidade dos limites obtidos por nossas heurísticas.

Parte dos resultados heurísticos deste trabalho estão publicados em [2], tendo sido fortalecidos ainda mais com a utilização de algoritmos exatos e com a inclusão de

técnicas de pré-processamento para o PPC, constituindo nossa principal contribuição para a resolução desse problema.

O trabalho está organizado como segue. No próximo capítulo definimos o PPC e apresentamos sua formulação clássica de programação inteira. Destacamos uma série de aplicações do PPC, que comprovam a importância do seu estudo. Em seguida, fazemos uma revisão bibliográfica dos principais trabalhos que abordam o PPC. No capítulo 3 abordamos as estratégias de resolução e os algoritmos que utilizamos para o PPC. No capítulo 4 descrevemos os resultados computacionais obtidos, comparando-os com resultados existentes na literatura. No capítulo 5 apresentamos uma conclusão e perspectivas de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Neste capítulo inicialmente definimos o Problema de Particionamento de Conjuntos (PPC) e apresentamos sua formulação clássica de programação inteira. Nas seções 2.2 e 2.3 abordamos problemas de cobertura relacionados ao PPC. Na seção 2.4 destacamos as principais aplicações do PPC. Na seção 2.5 fazemos uma revisão bibliográfica dos principais trabalhos que abordam esse problema.

2.1 Problema de Particionamento de Conjuntos

Seja $\mathcal{A} = [a_{ij}]_{m \times n}$ uma matriz de elementos 0–1, onde $\mathcal{M}$ e $\mathcal{N}$ representam, respectivamente, o conjunto de índices das linhas e das colunas de $\mathcal{A}$. Cada elemento da matriz é igual a um ($a_{ij} = 1$), se a coluna j cobrir a linha i, ou é igual a zero ($a_{ij} = 0$), caso contrário. A cada coluna j associamos um custo c_j.

Seja $\mathcal{J}(i) = \{j \mid a_{ij} = 1,\ j \in \mathcal{N}\}$ o conjunto de colunas que cobre a linha $i \in \mathcal{M}$ e $\mathcal{R}(j) = \{i \mid a_{ij} = 1,\ i \in \mathcal{M}\}$ o conjunto de linhas coberto pela coluna $j \in \mathcal{N}$.

Um particionamento é um conjunto de colunas $\mathcal{N}' \subseteq \mathcal{N}$, tal que toda linha de $\mathcal{M}$ é coberta exatamente por uma das colunas de $\mathcal{N}'$. A solução do problema consiste em encontrar um particionamento de colunas de custo mínimo. Formalmente, o PPC é

definido pelo seguinte modelo de programação inteira 0–1 [14]:

$$\min_{x \in \{0,1\}^n} \quad \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j = 1, \quad i = 1, \ldots, m \tag{2.2}$$

onde atribuímos o valor $x_j = 1$, se a coluna $j \in \mathcal{N}$ (com custo c_j) estiver na solução do problema, e $x_j = 0$, caso contrário.

2.2 Problema de Recobrimento de Conjuntos

O Problema de Recobrimento de Conjunto (PRC) é uma relaxação do PPC. A diferença entre eles reside exclusivamente na natureza das restrições dos problemas. O PRC é definido pela seguinte formulação de programação inteira 0–1 [30]:

$$\min_{x \in \{0,1\}^n} \quad \sum_{j=1}^{n} c_j x_j \tag{2.3}$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1, \ldots, m \tag{2.4}$$

Garfinkel e Nemhauser [26] abordam um modelo matemático que leva à transformação de um PPC em um PRC, enquanto [33] mostram a conversão entre esses problemas através de métodos algébricos.

2.3 Problema de *Packing*

O PPC, segundo [30], pode ser considerado um caso particular do Problema de *Packing* (PP), cuja fomulação matemática é definida como segue:

$$\max_{x \in \{0,1\}^n} \quad \sum_{j=1}^{n} c_j x_j \tag{2.5}$$

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j \leq 1, \quad i = 1, \ldots, m \tag{2.6}$$

O conjunto de restrições do PP condicionam a cobertura disjunta das linhas da matriz A por um subconjuntos de suas colunas. O modelo do PP possui grande aplicação prática na solução do problemas de cortes geométricos, acondicionamento de embalagens e carregamento de veículos. Técnicas de conversão de um PPC em um PP podem ser consultadas em [30].

2.4 Aplicações

Muitos problemas de otimização combinatória podem ser modelados como um PPC, o que justifica a variedade de áreas de aplicação desse problema, tais como:

- Alocação de serviços de emergência [43].
- Alocação de tripulação em linhas aéreas [32].
- Análise de amostras de sangue [38]
- Balanceamento de capacidade em linhas de produção [43].
- Caminho em Grafos [30].
- Distribuição do tráfego de comunicações em satélites [43].
- Estocagem de itens sujeitos à deterioração [30]
- Gestão estratégica [26].
- Investimento de capitais [43].
- Localização de facilidades [43].
- Lógica simbólica [17].
- Planejamento de cadeia de suprimentos [30].
- Planejamento de tarefas [30].
- Problema de coloração de grafos [30].
- Problema de corte [30].
- Projeto de circuitos [30].
- Recuperação de informações em banco de dados [5].
- Roteamento de petroleiros [23].
- Roteamento de veículos terrestres [13].

Devido à sua importância e ao seu impacto econômico, o problema de alocação de tripulação em linhas aéreas é provavelmente a área de aplicação mais estudada do PPC [20]. A variedade de aplicações do PPC reflete a importância do estudo desse problema, bem como a busca por metodologias eficientes para solucioná-lo.

2.5 Revisão Bibliográfica

Com relação ao problema de escalonamento de tripulação em linhas aéreas, citamos os seguintes trabalhos. Lavoie, Minoux e Odier [36] apresentam um algoritmo que aborda a alocação de tripulação como um problema de recobrimento de conjuntos e utiliza geração parcial de colunas para solucioná-lo. Desaulniers et al [18] utiliza um modelo de fluxo de redes com recursos variáveis formulado como um problema de programação inteira, não linear e multi-produto. O autor aplica uma extensão do princípio da decomposição de Dantzig-Wolfe e reduz o problema para um PPC que é resolvido pela aplicação de um esquema de *Branch and Bound*. Emden–Weinert e Proksch [19] apresentam uma abordagem para a programação de tripulações via recozimento simulado (*Simulated Annealing*). Os autores abordam como melhorar a performance dessa heurística no que diz respeito ao tempo de execução e à qualidade da solução final. Ozdemir e Mohan [39] modelam o problema de alocação de tripulações como um grafo, no qual os vértices são os vôos e as arestas representam restrições de dependência entre os vôos. Cada caminho no grafo é uma seqüência factível de vôos que pode ser associada a uma tripulação. O autor aplica a esse modelo um algoritmo genético e uma heurística de busca local. Pimentel [40] propõe uma nova metodologia para a solução de problemas de recobrimento de conjuntos de grande porte, com aplicação no problema de alocação de tripulações de companhias aéreas. A autora apresenta uma formulação que divide a solução do problema em duas etapas: geração de rotas e otimização. Na etapa de otimização é desenvolvida uma metodologia para resolver problemas modelados como um problema de recobrimento de conjuntos, enquanto na etapa de geração de rotas é utilizado um algoritmo paralelo, baseado na meta-heurística GRASP [41]. Beasley e Cao [11] aplicam um algoritmo baseado em programação dinâmica para o problema de escalonamento de tripulação em linhas aéreas.

Como exemplos de estratégias de resolução do PPC, citamos os seguintes trabalhos. Balas e Padberg [6] destacam as técnicas de enumeração implícita e métodos de planos de corte baseados no simplex para a resolução do PPC. Marsten [37] descreve as técnicas de *Branch-and-Bound* e de *Branch-and-Cut* como sendo as ferramentas, baseadas em programação linear, mais populares entre pesquisadores para solucionar o PPC. Fleuren [24] utiliza um esquema de *Branch-and-Bound* com relaxação lagrangeana para obter

limites inferiores para o PPC. Fisher e Kedia [23] apresentam um algoritmo que aplica heurísticas ao problema dual da relaxação linear para fornecer limites inferiores para um esquema de *Branch-and-Bound*. Hoffman e Padberg [32] propõem um algoritmo exato, baseado em *Branch-and-Cut* que envolve a relaxação linear do problema e a incorporação de cortes derivados do estudo do politopo do PPC. Atamtürk et al [4] apresentam um algoritmo heurístico que incorpora redução do problema, programação linear, planos de corte, procedimento dual lagrangeano e técnicas de perturbação dos custos. Chu e Beasley [15] desenvolvem uma heurística baseada em algoritmos genéticos. Borndörfer [12] desenvolve um algoritmo baseado em *Branch-and-Cut* que utiliza relaxação linear. Barahona e Anbil [8] utilizam um algoritmo de volume para resolver a relaxação linear do PPC, utilizando uma extensão do algoritmo de otimização pelo subgradiente, para alcançar soluções primais e duais de boa qualidade. Segundo os autores, esse procedimento é rápido e simples, requerendo menor esforço computacional quando associado ao método *simplex*, e pode ser paralelizado de forma direta. Klabjan [35] apresenta um algoritmo que computa função dual subaditiva (FSO) ótima para problemas de programação inteira. O autor mostra como gerar FSO geradora e como obter solução primal ótima para o PPC, utilizando pré-processamento, desigualdades de clique, heurística de expansão baseada em programação linear, procedimento de geração de linhas e um esquema de *Branch-and-Cut*. Desigualdades válidas para o PPC são encontradas em Balas e Padberg [6] e em Sherali e Lee [43].

No próximo capítulo, abordamos as técnicas de resolução heurísticas e exatas que utilizamos para resolver instâncias do PPC da literatura.

Capítulo 3

Estratégias de Resolução

Inicialmente, abordamos técnicas de pré-processamento para o Problema de Particionamento de Conjuntos (PPC). Na seção 3.2 apresentamos uma heurística gulosa construtiva e as estratégias de seleção de colunas que adotamos para a obtenção de soluções viáveis para o problema. Na seção 3.3 apresentamos um esquema de busca tabu para obtenção de limites superiores. Nas seções 3.4 e 3.5 abordamos, respectivamente, um esquema de relaxação lagrangeana e uma heurística de otimização pelo subgradiente para obtenção de limites inferiores. Nas seções 3.6 e 3.7 apresentamos heurísticas lagrangeanas, que integram as técnicas de resolução anteriores com o objetivo de melhorar os limites obtidos para o PPC. Na seção 3.8 descrevemos o algoritmo de integração utilizado. Na seção 3.9 abordamos o esquema de *Branch and Bound* que adotamos para tentar solucionar o PPC à otimalidade.

3.1 Técnicas de Pré-processamento

Esquemas de pré-processamento são procedimentos empregados na reformulação de modelos de programação matemática, que podem reduzir as dimensões do problema e o tempo de execução de algoritmos utilizados na sua resolução. São técnicas baseadas em implicações lógicas derivadas de condições de viabilidade e otimalidade [4].

Para o PPC, procedimentos de redução podem ser aplicados para tentar reduzir as dimensões do modelo de programação inteira, de modo que a solução ótima da instância do problema original seja a mesma da instância reformulada. Pela aplicação dessas técnicas, colunas redundantes do PPC que jamais poderiam fazer parte de uma solução viável são removidas. Isso pode, inclusive, resultar na eliminação de restrições redundantes que são automaticamente satisfeitas quando outras o são. A redução do número de variáveis e de restrições a serem consideradas faz com que alguns métodos de resolução trabalhem melhor [12]. No trabalho de Eso [21] é mostrado que a ordem em que diferentes procedimentos de redução são aplicados não influencia o

problema resultante, mas afeta o custo computacional envolvido no pré-processamento. Em Borndörfer [12], há uma listagem de métodos de pré-processamento e análises probabilísticas da usabilidade deles. Pesquisas recentes sobre o emprego dessas técnicas para o PPC podem ser encontradas em Savelsbergh [42] e Klabjan [35].

A seguir apresentamos as técnicas de pré-processamento que utilizamos para o PPC.

3.1.1 Colunas iguais

Se uma coluna $j \in \mathcal{N}$ puder ser representada por uma combinação de k outras colunas de menor custo total, tal que $k > 0$, então a coluna j pode ser removida do problema. Considerando $S \subset \mathcal{N} \setminus j$ como sendo uma coleção de k colunas, essa técnica de redução pode ser formalmente definida como segue, segundo [44]. Sejam j e S tais que

$$\mathcal{R}(j) = \bigcup_{h \in S} \mathcal{R}(h), \text{ onde } \mathcal{R}(h_1) \cap \mathcal{R}(h_2) = \emptyset, \ \forall \, h_1, h_2 \in S,$$

Se $c_j \geq \sum_{h \in S} c_h$ então a coluna $j \in \mathcal{N}$ pode ser removida do problema.

Utilizamos esse método de pré-processamento particularmente para $k = 1$, para eliminarmos toda coluna j de custo c_j que seja igual à coluna p de custo c_p tal que $j, p \in \mathcal{N}$ e $c_j \geq c_p$. Não implementamos esse procedimento para valores de $k > 1$, devido ao grande número de combinações de colunas a serem analisadas, que implicaria num maior custo computacional.

No exemplo abaixo, mostramos uma instância do PPC onde as colunas j_1 e j_6 são iguais. A coluna j_6 é eliminada, uma vez que $c_6 > c_1$.

	j_1	j_2	j_3	j_4	j_5	j_6
i_1	**1**	0	1	0	0	**1**
i_2	**0**	1	0	1	0	**0**
i_3	**1**	0	1	1	1	**1**
i_4	**0**	1	0	0	0	**0**
i_5	**1**	0	0	0	1	**1**
	$c_1 = 1$	$c_2 = 3$	$c_3 = 5$	$c_4 = 2$	$c_5 = 4$	$c_6 = 6$

3.1.2 Linhas dominantes e linhas iguais

Se existirem no PPC duas linhas (restrições) $p, q \in \mathcal{M}$, tais que $\mathcal{J}(p) \subseteq \mathcal{J}(q)$, então dizemos que a linha q "domina" a linha p. Todas as colunas que cobrem q e não cobrem

p podem ser eliminadas do problema, bem como a linha p, uma vez que os conjuntos $\mathcal{J}(p)$ e $\mathcal{J}(q)$ passam a ser iguais, após a eliminação das colunas de $\mathcal{J}(q) \setminus \mathcal{J}(p)$ [35].

A seguir mostramos um exemplo da aplicação desse procedimento numa instância do PPC. Na tabela 3.1, $\mathcal{J}(i_5) \subseteq \mathcal{J}(i_4)$, então podemos eliminar a coluna j_1, pois a atribuição dela à solução, $x_1 = 1$, violaria a restrição associada à linha i_5. Após a eliminação da coluna j_1, as linhas i_4 e i_5 tornam-se iguais e a linha i_5 pode ser eliminada. Na tabela 3.2, resultante após as eliminações na tabela 3.1, $\mathcal{J}(i_2) \subseteq \mathcal{J}(i_3)$. As colunas j_2 e j_5, que cobrem a linha i_3 mas não cobrem a linha i_2, podem ser eliminadas, bem como a linha "dominada" i_2, resultando na instância presente na tabela 3.3.

Tabela 3.1:

	j_1	j_2	j_3	j_4	j_5	j_6
i_1	0	1	1	0	0	1
i_2	1	0	0	1	0	1
i_3	0	1	0	1	1	1
i_4	1	0	1	1	0	0
i_5	0	0	1	1	0	0

Tabela 3.2:

	j_2	j_3	j_4	j_5	j_6
i_1	1	1	0	0	1
i_2	0	0	1	0	1
i_3	1	0	1	1	1
i_4	0	1	1	0	0

Tabela 3.3:

	j_3	j_4	j_6
i_1	1	0	1
i_3	0	1	1
i_4	1	1	0

3.1.3 Clique de linha

O nome desse método de pré-processamento se deve ao fato de que uma instância do PPC pode ser representada por um grafo G conexo não orientado [12], onde os vértices são as colunas $j \in \mathcal{N}$. Há uma aresta interligando um par de vértices de G quando eles cobrem pelo menos uma linha em comum. Portanto, cada restrição do PPC é uma clique em G e somente um vértice de cada clique pode estar presente numa solução viável do problema.

Este procedimento de redução é definido como segue, conforme [35]. Se existir no PPC uma linha i e uma coluna j, tais que $j \notin \mathcal{J}(i)$ e $\mathcal{R}(j) \cap \mathcal{R}(k) \neq \emptyset$, para todo $k \in \mathcal{J}(i)$, então podemos eliminar a coluna j.

A instância do PPC a seguir ilustra a aplicação desse procedimento para eliminação de colunas. Na tabela 3.4, com relação à linha i_1, temos o conjunto $\mathcal{J}(i_1) = \{j_1, j_2, j_6, j_7\}$ e seu complemento $\overline{\mathcal{J}(i_1)} = \{j_3, j_4, j_5\}$. A coluna $j_5 \in \overline{\mathcal{J}(i_1)}$ cobre pelo menos uma linha em comum com todas as colunas que cobrem a linha i_1. Conseqüentemente, a coluna j_5 pode ser eliminada do problema, uma vez que a atribuição dela a uma solução do PPC, $x_5 = 1$, violaria a restrição associada a linha i_1. A tabela 3.5 mostra a instância após a eliminação da coluna j_5. O mesmo procedimento é aplicado a todas as linhas da instância à procura de colunas para eliminação que satisfaçam as condições mencionadas.

Tabela 3.4:

	j_1	j_2	j_3	j_4	j_5	j_6	j_7
i_1	1	1	0	0	0	1	1
i_2	0	0	1	1	0	1	0
i_3	1	0	1	0	1	0	1
i_4	0	1	0	1	1	1	1

Tabela 3.5:

	j_1	j_2	j_3	j_4	j_6	j_7
i_1	1	1	0	0	1	1
i_2	0	0	1	1	1	0
i_3	1	0	1	0	0	1
i_4	0	1	0	1	1	1

A seguir apresentamos um algoritmo guloso e as estratégias de seleção de colunas adotadas para a obtenção de soluções viáveis, que posteriormente são utilizadas num procedimento de busca tabu para encontrar limites superiores para instâncias do PPC.

3.2 Heurística Gulosa Construtiva

Utilizamos uma heurística gulosa construtiva que utiliza critérios de seleção de colunas para obtenção de soluções primais viáveis para o PPC. Tais critérios provêm de heurísticas desenvolvidas por [5,16,45] para o problema de recobrimento de conjuntos.

A heurística gulosa funciona como segue. Inicialmente, nenhuma variável é representada no vetor solução, $x_j = 0,\ j \in \mathcal{N}$. Ordenamos as variáveis de acordo com o critério selecionado, dentre os apresentados em 3.2.1, 3.2.2 e 3.2.3. A cada iteração, selecionamos uma variável r para compor uma solução viável ($x_r = 1$). Como conseqüência, toda coluna $k \in \mathcal{J}(i) \backslash r$ tal que a linha $i \in \mathcal{R}(r)$ é eliminada do problema. Esse procedimento termina quando uma solução viável é encontrada ou quando verificamos que o critério de seleção de colunas adotado não obtém sucesso na determinação de uma solução viável. Cada um dos critérios de seleção de colunas é avaliado separadamente pela heurística e a solução viável do PPC com menor custo é utilizada, caso seja encontrada.

A seguir descrevemos os critérios de seleção de colunas implementados na heurística construtiva gulosa para o PPC.

3.2.1 Critério de seleção de Chvàtal

O critério de seleção de colunas de [16] consiste na escolha de uma coluna $j \in \mathcal{N}$ do PPC por iteração, para compor uma solução viável, de acordo com a razão entre o custo da coluna j, c_j, e a cardinalidade do conjunto $\mathcal{R}(j)$. As colunas que apresentam menor razão e não violam as restrições do problema têm preferência na seleção para compor uma solução viável para o PPC.

3.2.2 Critérios de seleção de Balas e Ho

Os critérios de seleção de [5] consistem na utilização das seguintes funções de avaliação de colunas:

- c_j: ordenação das colunas $j \in \mathcal{N}$ do problema pelo custo crescente, dando preferência às colunas de menor custo para compor o vetor solução.
- $\frac{c_j}{log_2(k_j)}$: ordenação das colunas $j \in \mathcal{N}$ do problema pela razão entre o custo da coluna j e o logaritmo do número de linhas cobertas na base 2.
- $\frac{c_j}{k_j\, log_2(k_j)}$: ordenação das colunas $j \in \mathcal{N}$ do problema pela razão entre o custo da coluna j e o número de linha cobertas, multiplicado pelo logaritmo na base 2 deste valor.
- $\frac{c_j}{k_j\, ln(k_j)}$: ordenação das colunas $j \in \mathcal{N}$ do problema pela razão entre o custo da coluna j e o número de linhas cobertas, multiplicado pelo logaritmo natural deste valor.

Nas fórmulas acima, c_j é o custo da coluna j e k_j é a cardinalidade do conjunto $\mathcal{R}(j)$.

3.2.3 Critérios de seleção de Vasko e Wilson

Além das funções de avaliação mencionadas, utilizamos também os critérios de seleção de colunas de [45]:

- $\frac{c_j}{k_j^2}$: ordenação das colunas $j \in \mathcal{N}$ do problema pela razão entre o custo da coluna j e o quadrado do número de linhas cobertas.
- $\frac{\sqrt{c_j}}{k_j^2}$: ordenação das colunas $j \in \mathcal{N}$ do problema pela razão entre a raiz quadrada do custo da coluna j e o quadrado do número de linhas cobertas.

e a seleção de colunas de acordo com o índice crescente que as identifica.

A solução encontrada pela heurística gulosa mencionada é parâmetro de entrada para a meta-heurística busca tabu, apresentada a seguir, que tem como objetivo obter limites superiores para o PPC.

3.3 Busca Tabu

A meta-heurística busca tabu (BT) ou *tabu search* foi proposta, independentemente, por [28] e por [31] e constitui um procedimento iterativo para resolver problemas de otimização combinatória. A BT possui mecanismos para escapar de ótimos locais, no intuito de tentar encontrar a solução ótima global.

De acordo com [29], a BT parte de uma solução viável inicial $s_0 \in S$, onde S constitui o conjunto de soluções viáveis do problema analisado. Em problemas de minimização como o PPC, a BT, a cada iteração, explora um subconjunto V da vizinhança $\mathcal{V}(s)$ da solução corrente s, em busca de um ótimo local através da seleção da solução vizinha $s' \in \mathcal{V}$ de menor valor, mesmo que o valor de s' seja maior que o de s. A transição entre as soluções $s \rightarrow s'$, que ocorre a cada iteração, constitui o que denominamos de *movimento*. Uma função de avaliação f é utilizada para avaliar a qualidade da solução corrente s. A melhor solução gerada até a iteração corrente, s^*, é sempre armazenada.

Para prevenir possíveis retornos a soluções previamente geradas (ciclagem), a BT utiliza uma memória de curto prazo ou *lista tabu*, que identifica os movimentos mais recentemente realizados e os impede de serem executados por um determinado número $|T|$ de iterações.

Movimentos presentes na *lista tabu* podem ser realizados, caso a nova solução $s' \in \mathcal{V}(s)$ satisfaça o *critério de aspiração*, que permite a realização de um movimento tabu se ele levar a uma solução cujo valor seja menor do que o da melhor solução encontrada até então, s^*.

A *lista tabu* constitui uma memória de curto prazo que possibilita uma diversificação na busca por novas soluções, à medida que direciona essa busca para novas áreas do espaço de solução, permitindo a fuga de ótimos locais. Memórias de médio e longo prazo também são utilizadas para armazenar soluções completas e informações de suas vizinhanças. O uso desses mecanismos de memória possibilita estratégias de aprendizado, que podem ser utilizadas para a realização de buscas mais minuciosas sobre uma região promissora do espaço de busca (*intensificação*). O armazenamento dos movimentos e das características históricas das soluções alcançadas leva a uma análise profunda da vizinhança da solução corrente e guia a busca para regiões inexploradas do espaço de soluções (*diversificação*).

O algoritmo 3.1 mostra o funcionamento da busca tabu que adotamos para o PPC. Os parâmetros de entrada são definidos como segue. f é uma função de avaliação de soluções. h é a função de aspiração que avalia se o melhor movimento da iteração corrente, $Iter$, está na *lista tabu* e determina a sua realização se a nova solução gerada tiver custo inferior ao da melhor solução conhecida. $|\mathcal{V}|$ indica a cardinalidade do conjunto de soluções existentes na vizinhança $\mathcal{V}(s)$ da solução corrente s a cada iteração. BT_{max} indica o número máximo de iterações sem melhoria no custo da melhor solução viável encontrada, que constitui o critério de parada adotado na nossa busca tabu. $\mathcal{A}$ e c são, respectivamente, a matriz de restrições e o vetor de custos das colunas do PPC. Já s constitui uma solução viável inicial que obtemos a partir da heurística construtiva

gulosa e dos critérios de seleção de colunas abordados na seção 3.2, enquanto p é um parâmetro percentual utilizado na análise da vizinhança da solução corrente s (busca local).

Algoritmo 3.1 Algoritmo Busca Tabu

Entrada: f, h, $|\mathcal{V}|$, $|T|$, BT_{max}, $\mathcal{A} = [a_{ij}]_{m \times n}$, $c \in \mathbb{R}^+$, s, p
Saída: Limite superior s

1: $s^* \leftarrow s$; /* solução inicial */
2: $Iter \leftarrow 0$; /* iteração corrente */
3: $MelhorIter \leftarrow 0$;
4: $T \leftarrow \emptyset$; /* lista tabu */
5: **enquanto** Condição de parada não é satisfeita **fazer**
6: $Iter \leftarrow Iter + 1$;
7: Realizar busca local para obter o melhor movimento s' da vizinhança da solução corrente, $\mathcal{V}(s)$, tal que s' não seja *tabu* ($s' \notin T$) ou $f(s') < h(f(s))$;
8: Atualizar a lista tabu;
9: $s \leftarrow s'$; /* realiza movimento */
10: **se** $f(s) < f(s^*)$ **então**
11: $s^* \leftarrow s$;
12: $MelhorIter \leftarrow Iter$;
13: **fim se**
14: Atualizar a função de aspiração h;
15: **fim enquanto**
16: $s \leftarrow s^*$;
17: Retorne s;

A BT, a cada iteração, seleciona o melhor movimento $s \rightarrow s'$, que não seja tabu ou que satisfaça o critério de aspiração. Na atualização da *lista tabu*, os movimentos mais recentes são armazenados e impedidos de serem realizados durante $|T|$ iterações, exceto se o critério de aspiração for satisfeito, e depois são retirados da lista tabu.

A melhor solução s' escolhida é aquela com melhor avaliação na vizinhança da solução corrente s, com relação ao valor da função objetivo, à lista tabu e ao critério de aspiração. Realizamos uma busca local na vizinhança $\mathcal{V}(s)$ como segue. A solução corrente s é formada por um subconjunto de colunas de $\mathcal{A}$ que constitui uma solução viável para o PPC. Seja $C(|s|, 2)$ o total de combinações de pares de colunas de s dois a dois. Seja $\bar{s} = \mathcal{N} \backslash s$. Selecionamos aleatoriamente $p\%$ dos pares de colunas da solução corrente s que são utilizados na busca por trocas entre colunas de s e de $\bar{s}$, onde p é um parâmetro de entrada. Uma coluna de s pode ser trocada por duas de $\bar{s}$ ou duas colunas de s podem ser trocadas por uma ou duas colunas de $\bar{s}$, desde que as colunas envolvidas nas possíveis trocas cubram o mesmo conjunto de linhas. Após a coleta de todos os movimentos possíveis, a BT seleciona aquele que apresenta maior descréscimo no custo da solução corrente ou menor acréscimo, caso não haja

trocas que proporcionem redução no custo da solução corrente s. Dessa forma, os movimentos promovem uma transição entre soluções viáveis do PPC a cada iteração da BT, em busca de um ótimo global.

A melhor solução obtida pela BT constitui um limite superior para o custo da solução ótima do PPC e é utilizado como parâmetro de entrada para a heurística de otimização pelo subgradiente e para as heurísticas lagrangeanas definidas nas próximas seções.

3.4 Relaxação Lagrangeana

Uma alternativa à utilização da Relaxação de Programação Linear para o PPC é a Relaxação Lagrangeana (RL) que consiste na dualização das restrições de igualdade do modelo de programação inteira, incorporando-as à função objetivo através de multiplicadores de Lagrange. A solução ótima do problema de programação inteira resultante (problema lagrangeano) fornece um limite inferior do valor da solução ótima do PPC. Com base no limite inferior encontrado e num limite superior conhecido para o custo da solução do problema podemos avaliar quão próximo da solução ótima está uma solução viável disponível. De acordo com [27], experiências práticas têm indicado que a RL fornece muito bons limites inferiores com custo computacional razoável para problemas de otimização combinatória.

Considere a formulação de programação inteira descrita em (2.1)–(2.2). Associando ao conjunto de restrições (2.2) multiplicadores de Lagrange $\lambda \in \mathbb{R}^m$ e, em seguida, incorporando esse conjunto de restrições à função objetivo (2.1), obtemos a formulação do problema relaxado:

$$\mathrm{L}(\lambda) \;=\; \min_{x\in\{0,1\}^n} \quad \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \lambda_i \left(1 - \sum_{j=1}^{n} a_{ij} x_j\right) \tag{3.1}$$

que é equivalente a

$$\mathrm{L}(\lambda) \;=\; \min_{x\in\{0,1\}^n} \quad \sum_{j=1}^{n} \left(c_j - \sum_{i=1}^{m} \lambda_i\, a_{ij}\right) x_j + \sum_{i=1}^{m} \lambda_i \tag{3.2}$$

O *problema dual lagrangeano*, cuja solução ótima fornece o valor dos multiplicadores de lagrange que levam ao limite inferior máximo para o PPC, é dado por:

$$\max_{\lambda\in\mathbb{R}^m} \mathrm{L}(\lambda) = \max_{\lambda\in\mathbb{R}^m} \left[\min_{x\in\{0,1\}^n} \{c^t x + (\lambda)^t (e - \mathrm{A}x)\} \quad \right] \tag{3.3}$$

onde $e \in 1^m$ e $c \in \mathbb{R}^n_+$ é o vetor cuja componente c_j é o custo da coluna j.

O custo reduzido de uma coluna j do PPC é definido como:

$$C_j = c_j - \sum_{i=1}^{m} \lambda_i \, a_{ij}, \qquad j = 1, \ldots, n \tag{3.4}$$

A solução $\bar{x} \in \{0,1\}^n$ do problema lagrangeano, dado $\bar{\lambda} \in \mathbb{R}^m$, para $j = 1, \ldots, n$, é dada por:

$$\bar{x}_j = \begin{cases} 1, & \text{se } C_j \leq 0 \\ 0, & \text{caso contrário;} \end{cases} \tag{3.5}$$

e o valor correspondente da solução do problema lagrangeano (limite inferior para o PPC) é dado por:

$$Z_{LB} = \sum_{j=1}^{n} C_j \bar{x}_j + \sum_{i=1}^{m} \bar{\lambda}_i \tag{3.6}$$

A seguir apresentamos o método de otimização pelo subgradiente, aplicado para encontrar valores numéricos para os *multiplicadores de Lagrange* para o PPC.

3.5 Heurística de Otimização pelo Subgradiente

Inicialmente definimos subgradiente para o conjunto de restrições do PPC. Em seguida abordamos o mecanismo de funcionamento iterativo da otimização pelo subgradiente.

3.5.1 Subgradiente

Por definição, um vetor $g = (g_1, g_2, \ldots, g_m)^t$ é subgradiente de L em λ se, para todo $\overline{\lambda} \in \mathbb{R}^m$, temos que $\mathrm{L}(\overline{\lambda}) \leq \mathrm{L}(\lambda) + (\overline{\lambda} - \lambda)^t \, g$ [22].

O conjunto de subgradientes de L em λ é um conjunto convexo denotado por $\nabla \mathrm{L}(\lambda)$ e chamado de *subdiferencial* de L em λ. Em decorrência disso, se os vetores g^1 e g^2 são dois subgradientes em λ, então qualquer combinação convexa deles, $\theta \, g^1 + (1 - \theta) \, g^2$, para $\theta \in [0,1]$, também é um subgradiente.

Proposição 1 *Para $\lambda \in \mathbb{R}^m$, sejam $\mathcal{S}$ o conjunto de soluções do PPC e $H(\lambda) = \{\hat{x} \in \mathcal{S} \mid \mathrm{L}(\lambda) = \lambda^t \, e + (c^t - \lambda^t \, \mathcal{A}) \, \hat{x}\}$. Para todo $\hat{x} \in H(\lambda)$, $(e - \mathcal{A}\hat{x}) \in \nabla \mathrm{L}(\lambda)$. Logo $(e - \mathcal{A}\hat{x})$ é um subgradiente de $\mathrm{L}(\lambda)$ [22].*

Demonstração:

$\mathrm{L}(\lambda) = min \ \{\lambda^t e + (c^t - \lambda^t \mathcal{A}) x\} \ \leq \ \lambda^t e + (c^t - \lambda^t \mathcal{A}) x, \qquad \forall \, x \in \{0,1\}^n$.
Para um dado $\overline{\lambda} \in \mathbb{R}^n$, e em particular para $\hat{x} \in H(\lambda)$, temos que: $\mathrm{L}(\overline{\lambda}) \ \leq \ \overline{\lambda}^t e + (c^t - \overline{\lambda}^t \mathcal{A}) \, \hat{x}$. *Por definição de H(λ),* $\mathrm{L}(\lambda) = \lambda^t e + (c^t - \lambda^t \mathcal{A}) \, \hat{x}$.

Segue por subtração que

$\mathrm{L}(\overline{\lambda}) - \mathrm{L}(\lambda) \leq (\overline{\lambda}^t - \lambda^t)\, e - (\overline{\lambda}^t - \lambda^t)\, \mathcal{A}\hat{x} \leq (\overline{\lambda}^t - \lambda^t)\, (e - \mathcal{A}\hat{x}) \leq (\overline{\lambda}^t - \lambda^t)\, g, \quad \forall\, \lambda \in \mathbb{R}^m.$
Logo $g = (e - \mathcal{A}\hat{x})$ *constitui um subgradiente de* $\mathrm{L}(\lambda)$. *Portanto* $g \in \nabla \mathrm{L}(\lambda)$.

■

Logo, o subgradiente para cada restrição relaxada do PPC é dado por

$$g_i = 1 - \sum_{j=1}^{n} a_{ij} x_j, \qquad i = 1, \ldots, m \tag{3.7}$$

3.5.2 Otimização pelo subgradiente

A otimização pelo subgradiente é um processo iterativo que consiste na determinação de uma seqüência de multiplicadores de Lagrange, que se espera convergir para um vetor ótimo $\lambda^* \in \mathbb{R}^m$, solução do problema dual lagrangeano (3.3). A heurística de otimização pelo subgradiente para o PPC é definida como segue, segundo [10].

Partimos de um conjunto de multiplicadores $\lambda^0 \in \mathbb{R}^m$, tal que $\lambda^0 = 0$. A cada iteração $t \geq 0$ atualizamos os multiplicadores de Lagrange de acordo com a expressão:

$$\lambda_i^{t+1} = \lambda_i^t + tp^t\, g_i^t, \qquad i = 1, ..., m \tag{3.8}$$

onde g_i^t, a i-ésima componente do subgradiente g na iteração t, é dada por:

$$g_i^t = 1 - \sum_{j=1}^{n} a_{ij} \bar{x}_j, \quad i = 1, \ldots, m \tag{3.9}$$

sendo $\bar{x}$ a solução do problema lagrangeano 3.1 e 3.2. tp^t, o tamanho do passo na iteração t, é dado por:

$$tp^t = \pi \left(\frac{Z_{UB} - Z_{LB}^t}{\sum_{i=1}^{m} (g_i^t)^2} \right) \tag{3.10}$$

onde $0 < \pi \leq 2$. Z_{UB} é o melhor limite superior para o problema, obtido pela busca tabu, e Z_{LB}^t é o valor da solução do problema lagrangeano na iteração t.

Após um número máximo de iterações sem acréscimo no valor do melhor limite inferior conhecido, o parâmetro π é reduzido pela metade. Os critérios de parada são $Z_{UB} - Z_{LB}^t < 1$ ou $\pi < 0,005$.

A seguir apresentamos as heurísticas lagrangeanas.

3.6 Heurística Lagrangeana Pura

A *heurística lagrangeana pura* (HLP) consiste em associar às colunas do PPC o custo reduzido obtido nas iterações da otimização pelo subgradiente [3]. Aplicamos a heurística gulosa construtiva e a busca tabu às instâncias do PPC com os custos modificados, para tentar encontrar soluções viáveis de melhor qualidade. Quando a HLP obtém um limite superior inferior ao melhor limite superior corrente, Z_{UB} é atualizado. A HLP é executada nas iterações da otimização pelo subgradiente em que o valor da solução do problema lagrangeano é maior do que o melhor limite inferior corrente.

3.7 Heurística Lagrangeana de Custo Complementar

A diferença entre a HLP e a *heurística lagrangeana de custo complementar* (HLCC) é que aquela utiliza como custo da coluna j o valor C_j (custo reduzido), enquanto esta utiliza o valor $c_j(1 - \bar{x}_j)$, para $j = 1, 2, \ldots, n$, onde c_j é o custo original da coluna j e $\bar{x}_j$ é o valor da componente j na solução do problema lagrangeano [3]. Da mesma forma que na HLP, a heurística gulosa e a busca tabu são aplicadas às instâncias do PPC com os custos das colunas modificados. O objetivo é utilizar informações da estrutura da solução do problema lagrangeano, para tentar encontrar bons limites inferiores e superiores para as instâncias do PPC. A HLCC é executada nas iterações da otimização pelo subgradiente em que o valor da solução do problema lagrangeano é maior do que o melhor limite inferior corrente.

A seguir apresentamos algoritmos que combinam as técnicas discutidas nas seções anteriores para compor as heurísticas lagrangeanas para o PPC.

3.8 Algoritmo de Integração

As técnicas de redução e os métodos heurísticos de construção de soluções viáveis para o PPC são aplicados conforme o algoritmo 3.2.

Inicialmente as técnicas de pré-processamento são aplicadas para reduzir as dimensões do PPC. Em seguida, aplicamos os métodos heurísticos de construção de soluções viáveis. Avaliamos cada um dos critérios gulosos de seleção de colunas, descritos em 3.2.1, 3.2.2 e 3.2.3, em procedimentos distintos e utilizamos a solução de menor custo obtida, $X_{inicial}$, como parâmetro de entrada para a busca tabu, que procura obter uma nova solução viável com menor custo. O custo da melhor solução viável obtida constitui o limite superior Z_{UB} que é retornado pelo algoritmo 3.2.

A otimização pelo subgradiente, associada às heurísticas lagrangeanas, é descrita pelo algoritmo 3.3. $\overline{Z}_{UB}$ e $\overline{Z}_{LB}$ constituem os melhores limites superior e inferior

Algoritmo 3.2 Técnicas de redução e construção de solução viável para o PPC

Entrada: Matriz $\mathcal{A} = [a_{ij}]_{m \times n}$ de elementos 0–1, vetor de custos $c \in \mathbb{R}^n_+$
Saída: Limite superior Z_{UB}

1: /* Pré-processamento */
2: Aplicação do procedimento para colunas iguais
3: **repetir**
4: Aplicação dos procedimentos para linhas dominantes e linhas iguais
5: Aplicação do procedimento para clique de linha
6: **até** não haver eliminação de linhas ou colunas
7: /* Métodos heurísticos de construção */
8: Executar procedimentos de construção de soluções viáveis, a partir dos critérios gulosos de seleção de colunas. $X_{inicial}$ constitui a melhor solução encontrada com custo c_X
9: /* Busca Tabu */
10: Executar a busca tabu que recebe a solução $X_{inicial}$ e retorna o limite superior Z_{UB}

obtidos, respectivamente. t indica a iteração corrente, *cont* é um contador para a quantidade de iterações sem melhoria no valor de $\overline{Z}_{LB}$. O valor de π é reduzido pela metade, quando *cont* atinge o valor máximo α, que indica a quantidade limite de iterações sem melhoria no valor do melhor limite inferior corrente. Nas iterações da otimização pelo subgradiente em que o valor da solução do problema lagrangeano é maior do que $\overline{Z}_{LB}$, aplicamos a HLP ou a HLCC, para tentar reduzir o GAP entre $\overline{Z}_{UB}$ e $\overline{Z}_{LB}$ do PPC.

De acordo com o algoritmo 3.3, na primeira iteração da otimização pelo subgradiente, atribuímos ao vetor de multiplicadores de Lagrange o valor $\lambda = 0$. Inicialmente o parâmetro de relaxação π assume o valor 2.

Nas iterações da otimização pelo subgradiente, quando um dos componentes do vetor de multiplicadores de Lagrange λ_i, para $i = 1, \ldots, m$, for nulo e o subgradiente correspondente for negativo, então realizamos a projeção do valor do subgradiente para zero. Segundo [10] essa projeção é necessária para que o fator do subgradiente não interfira no cálculo do tamanho do passo, o que favorece a convergência para λ^* na otimização pelo subgradiente.

Os critérios de parada adotados foram valor de $\pi < 0.005$ ou $Z_{UB} - Z_{LB} < 1$. A seguir abordamos o esquema de *Branch and Bound* que adotamos para o PPC.

3.9 *Branch and Bound*

O *Branch and Bound* (B&B) realiza um particionamento iterativo do espaço de soluções $\mathcal{S}$ do PPC em subproblemas, tal que $\mathcal{S}$ é definido pelo conjunto de restrições $\mathcal{A}x = 1$ e $x \in \{0, 1\}^n$ da formulação de programação inteira do PPC.

Algoritmo 3.3 Otimização pelo subgradiente e heurísticas lagrangeanas

Entrada: Matriz $\mathcal{A} = [a_{ij}]_{m \times n}$, vetor de custos $c \in \mathbb{R}^n_+$, limite superior Z_{UB}, α.
Saída: Limite superior $\overline{Z}_{UB}$, limite inferior $\overline{Z}_{LB}$.

1: $t \leftarrow 0$; $\pi \leftarrow 2$; $\lambda_i^t \leftarrow 0$; $g_i^t \leftarrow 1$, $i = 1, \ldots, m$; $\overline{Z}_{UB} \leftarrow Z_{UB}$;
2: $C_j^t \leftarrow c_j$, $j = 1, \ldots, n$; $Z_{LB}^t \leftarrow 0$; $\overline{Z}_{LB} \leftarrow 0$; tp^t é dado pela equação (3.10); $cont \leftarrow 0$;
3: **repetir**
4: Atualizar multiplicadores de Lagrange λ_i^{t+1}, $i = 1, \ldots, m$, por (3.8);
5: Calcular custo reduzido C_j^{t+1}, $j = 1, \ldots, n$, por (3.4);
6: Obter solução $\bar{x}$ do problema lagrangeano por (3.5);
7: Calcular subgradiente g_i^{t+1}, $i = 1, \ldots, m$, por (3.9);
8: **se** $g_i^{t+1} < 0$ e $\lambda_i^{t+1} = 0$ **então**
9: $g_i^{t+1} \leftarrow 0$;
10: **fim se**
11: Calcular custo do problema lagrangeano Z_{LB}^{t+1} por (3.6);
12: **se** $Z_{LB}^{t+1} > \overline{Z}_{LB}$ **então**
13: $\overline{Z}_{LB} \leftarrow Z_{LB}^{t+1}$;
14: $cont \leftarrow 0$;
15: Executar heurística lagrangeana, que retorna limite superior Z_{UB}; /* HLP ou HLCC */
16: **se** $Z_{UB} < \overline{Z}_{UB}$ **então**
17: $\overline{Z}_{UB} \leftarrow Z_{UB}$;
18: **fim se**
19: **senão**
20: $cont \leftarrow cont + 1$;
21: **se** $cont = \alpha$ **então**
22: $\pi \leftarrow \pi/2$;
23: $cont \leftarrow 0$;
24: **fim se**
25: **fim se**
26: Calcular tamanho do passo tp^{t+1} por (3.10);
27: $t \leftarrow t + 1$;
28: **até** ($Z_{UB} - Z_{LB} < 1$ ou $\pi < 0.005$)

Utilizamos a relaxação linear para encontrar limites inferiores nos nós da árvore de B&B, sendo $\mathcal{S}_R = \{x \in \mathbb{R}^n \mid \mathcal{A}x = 1,\ 0 \leq x \leq 1\}$ o conjunto de soluções do problema relaxado.

O funcionamento do B&B para um problema de minimização é apresentado pelo algoritmo 3.4, onde L denota a lista de subproblemas (nós) ativos $\{\text{NO}^i\}$ e NO^0 é o nó raiz, ou seja, é a relaxação linear do problema inteiro original. z_i denota o limite inferior correspondente ao valor da relaxação linear do subproblema NO^i. z_{ub} denota o valor da melhor solução inteira viável conhecida do problema original. z_{lb} denota o valor do melhor limite inferior obtido através da relaxação linear dos nós gerados na árvore de B&B. x_i denota a solução da relaxação linear de NO^i. Inicialmente, z_{ub} assume o valor de Z_{UB}, melhor limite superior obtido pela execução das heurísticas lagrangeanas.

O algoritmo B&B é tipicamente visto como uma árvore de busca. A árvore é construída iterativamente através da formação de novos nós por um processo de ramificação (*branching*) de um nó k existente, para o qual a solução ótima da relaxação é fracionária. Nesse processo dois nós filhos são formados a partir da seleção de uma das variáveis x_j de valor fracionário (variável de *branching*). Os nós filhos são criados a partir da fixação de x_j em 1 e em 0, respectivamente, no intuito de assegurar que o conjunto de restrições associado aos nós filhos de k não inclua soluções nas quais a variável de *branching* escolhida assuma o mesmo valor fracionário.

A etapa denominada poda é utilizada como critério para avaliar se um nó da árvore não necessita ser explorado. Classificamos um nó como *fathoming* nas seguintes situações:

- Inviabilidade: quando pelo menos uma das restrições do modelo de programação linear é violada.
- *Bound*: quando o valor da sua relaxação linear é maior ou igual ao valor da melhor solução inteira viável conhecida (z_{ub});
- quando o valor da relaxação linear do nó é inteiro e esse valor é inferior z_{ub}. Nesse caso, atualizamos z_{ub} para assumir o valor z_i e removemos de L todos os subproblemas j com $z_j \geq z_{ub}$.

Realizamos a poda dos nós considerados *fathoming* para tentar obter a solução ótima e provar a otimalidade sem uma busca exaustiva no espaço de soluções do problema.

O *B&B* termina quando todos os nós são *fathoming* ou quando encontramos um nó da árvore cuja solução é inteira e tem valor (limite inferior) igual ao valor do melhor limite superior conhecido, z_{ub}. Neste caso, ocorre a prova de que z_{ub} constitui o valor de uma solução ótima do PPC.

Algoritmo 3.4 *Branch-and-Bound*

1: /* Inicialização */
   L $\leftarrow \{\text{NO}^0\}$;
   $z_0 \leftarrow 0$;
   $z_{lb} \leftarrow 0$;
   $z_{ub} \leftarrow Z_{UB}$;
   $opt \leftarrow 0$;
2: /* Condições de parada */
3: **enquanto** $L \neq \emptyset$ ou $opt = 0$ **fazer**
4: Selecionar e remover um subproblema NO^i de L, tal que $i \geq 0$;
5: Computar a relaxação linear de NO^i, cujo valor é z_i;
6: **se** $z_i > z_{ub}$ **então**
7: Ir para passo 3; /* poda */
8: **fim se**
9: **se** $z_i > z_{lb}$ **então**
10: $z_{lb} \leftarrow z_i$;
11: **fim se**
12: **se** x_i for uma solução inteira **então**
13: **se** $z_i < z_{ub}$ **então**
14: $z_{ub} \leftarrow z_i$;
15: Remover de L todos os subproblemas j com $z_j \geq z_{ub}$; /* poda */
16: **fim se**
17: **se** $z_{ub} = z_{lb}$ **então**
18: $opt \leftarrow 1$;
19: **fim se**
20: **senão**
21: /* Particionamento */
Seja $\{S^{ij}\}_{j=1}^{j=2}$ um particionamento do conjunto de restrições S^i do problema $\{\text{NO}^i\}$. Adicionar os problemas $\{\text{NO}^{ij}\}_{j=1}^{j=2}$ a L, se forem viáveis, onde $\{\text{NO}^{ij}\}$ é NO^i com região de viabilidade restrita a $\{S^{ij}\}$;
22: **fim se**
23: **fim enquanto**
24: **se** $opt = 1$ **então**
25: x^*, referente à melhor solução inteira viável, é solução ótima com valor z_{ub};
26: **senão se** existir x, referente à melhor solução inteira viável **então**
27: a solução x é retornada com valor z_{ub};
28: **senão**
29: o problema é inviável;
30: **fim se**

Selecionamos a variável mais fracionária como critério de seleção de variáveis de *branching*. Como critério de seleção de nós adotamos o *best bound*, que consiste em escolher, entre todos os nós inexplorados de L, aquele que apresenta menor valor de limite inferior.

Para implementar a lista de nós não explorados L, utilizamos a estrutura de dados *heap* na qual os nós ficam dispostos em um vetor e o nó de menor limite inferior fica situado na cabeça do *heap*, tendo maior prioridade na seleção do próximo nó a ser analisado pelo B&B.

A seguir abordamos os resultados computacionais obtidos pelas estratégias de resolução apresentadas.

Capítulo 4

Resultados Computacionais

Neste capítulo abordamos os resultados computacionais alcançados pelas estratégias de resolução apresentadas no capítulo anterior. Inicialmente, descrevemos as instâncias do PPC que utilizamos neste trabalho e o resultado da aplicação das técnicas de pré-processamento sobre elas. Na seção 4.3 apresentamos os resultados da execução da rotina *CPXmipopt* da *Cplex Callable Library* para resolver instâncias do PPC. Na seção 4.4 descrevemos os resultados da meta-heurística busca tabu. Na seção 4.5 apresentamos os limites superior e inferior alcançados pelas heurísticas lagrangeanas. Na seção 4.6 abordamos os resultados alcançados pelo *Branch and Bound*. Na seção 4.7 comparamos os resultados obtidos pelas heurísticas lagrangeanas com resultados existentes na literatura.

4.1 Introdução

As técnicas de pré-processamento, as heurísticas construtiva gulosa e lagrangeanas e o esquema de *Branch and Bound* apresentados foram implementados com a linguagem de programação C++ no sistema operacional GNU Linux. Para computar a relaxação linear no esquema de *Branch and Bound*, utilizamos o Cplex [1]. A máquina utilizada para a realização dos testes dos algoritmos possui um processador Pentium IV de 3000 MHz e 512 MB de memória RAM DDR2.

Os algoritmos implementados foram utilizados para resolver 54 instâncias de Hoffman e Padberg [32] para o PPC, presentes na *OR-Library* [9]. O valor da densidade, relação entre a quantidade de elementos $a_{ij} = 1$ e o total de elementos da matriz $\mathcal{A}$, característica do PPC, varia entre 0,99% e 36,90%, o que mostra o quanto as instâncias do PPC utilizadas são espaças.

Em virtude de limitação de recurso de memória física, não fomos capazes de resolver a instância do PPC denominada *us01*, que possui um total de 145 linhas e 1053137 colunas. Por esse motivo, essa instância não está incluída nas tabelas dos resultados

computacionais alcançados neste trabalho.

Nas tabelas que seguem utilizamos a seguinte legenda:

instância	identificador das instâncias de [32]
linhas	número de linhas da instância do PPC anterior à aplicação do pré-processamento
colunas	número de colunas da instância do PPC anterior à aplicação do pré-processamento
linhas PP	número de linhas da instância do PPC após aplicação do pré-processamento
colunas PP	número de colunas da instância do PPC após aplicação do pré-processamento
% linha	percentual do número de linhas eliminadas no pré-processamento
% colunas	percentual do número de colunas eliminadas no pré-processamento
pClique	parâmetro utilizado na técnica de redução clique de linhas
custo ótimo	custo da solução ótima obtido pelo MIP Cplex
SIG	valor da solução da heurística gulosa construtiva
% SIG	diferença relativa entre SIG e o custo ótimo
LS BT	limite superior obtido pela busca tabu
% LS BT	diferença relativa entre LS tabu e o custo ótimo
iter BT	número de iterações executadas na busca tabu
LS HL	limite superior obtido pelas heurísticas lagrangeanas
% LS HL	diferença relativa entre LS HL e o custo ótimo
LI HL	limite inferior obtido pelas heurísticas lagrangeanas
% LI HL	diferença relativa entre LI HL e o custo ótimo
GAP	diferença relativa entre LS HL e LI HL
nIOS	número de iterações executadas da otimização pelo subgradiente
estimativa LS	estimativa de limite superior
% estimativa LS	diferença relativa entre estimativa LS e o custo ótimo
iter MIP	número de iterações do simplex
nNodes	número de nós usado para resolver o problema inteiro do PPC
nCortes	número de cortes de clique adicionados durante a otimização pelo Cplex
nDesigualdades	número de desigualdades de clique geradas pelo Cplex
LS Inic	limite superior inicial para o Branch and Bound
LI Root	limite inferior obtido na otimização do nó raiz do Branch and Bound
sol BB	solução obtida pelo Branch and Bound
gerados	número de nós gerados durante execução do Branch and Bound
expl	número de nós explorados durante execução do Branch and Bound
inexpl	número de nós inexplorados durante execução do Branch and Bound
t(s)	tempo de execução (CPU) em segundos

4.2 Pré-processamento

As técnicas de pré-processamento foram aplicadas conforme o algoritmo 4.1. Inicialmente, aplicamos o procedimento de eliminação de colunas iguais e, em seguida, de forma iterativa, aplicamos os procedimentos de linhas dominantes, linhas iguais e clique de linhas, até não haver mais eliminações de linhas e de colunas.

Algoritmo 4.1 Pré-processamento para o PPC

Entrada: Matriz A $= [a_{ij}]_{m \times n}$, vetor de custos $c \in \mathbb{R}^n_+$, *pClique*
Saída: Matriz A $= [a_{ij}]_{m \times n}$ com quantidade menor ou igual de linhas e de colunas

1: Aplicar procedimento de colunas iguais
2: **repetir**
3: Aplicar procedimento de linhas dominantes e iguais
4: Aplicar procedimento de clique de linha
5: **até** não haver eliminação de linhas ou colunas

A tabela 4.1 apresenta o número de linhas e de colunas da matriz $\mathcal{A} = [a_{ij}]_{m \times n}$ das instâncias do PPC antes e depois da aplicação dos métodos de pré-processamento, bem como a percentagem de eliminação desses elementos em relação ao total. O parâmetro de entrada *pClique* é determinado empiricamente para o procedimento *clique de linha* (3.1.3) e indica a quantidade máxima de colunas que devem cobrir uma linha para que esta seja considerada no processo de busca por colunas que podem ser eliminadas. De acordo com testes experimentais realizados, a aplicação das técnicas de pré-processamento com valor de parâmetro *pClique* maior que o informado resulta na mesma quantidade de linhas e de colunas eliminadas, mas pode exigir maior custo computacional. O valor de *pClique* igual a zero indica que o procedimento *clique de linhas* não elimina linhas ou colunas nas instâncias consideradas.

Os resultados computacionais apresentados na tabela 4.1 mostram que a aplicação dos métodos de pré-processamento na maioria das instâncias do PPC analisadas promovem reduções significativas no número de variáveis (colunas) e de restrições (linhas) do problema. Em média 5,91% das linhas e 27,35% das colunas são eliminadas na aplicação das técnicas de pré-processamento. Destacamos os resultados obtidos em *us02* onde houve a eliminação de 55% das linhas e em *us04*, onde houve redução de 84,98% das colunas.

Tabela 4.1 – Instâncias de Hoffman e Padberg [32]

instância	linhas	colunas	linhas PP	% linhas	colunas PP	% colunas	pClique	t(s)
aa01	823	8904	617	25,03	7544	15,27	308	6,78
aa02	531	5198	362	31,83	3848	25,97	179	1,84
aa03	825	8627	553	32,97	6707	22,26	176	6,76
aa04	426	7195	343	19,48	6123	14,90	144	1,98
aa05	801	8308	533	33,46	6246	24,82	167	4,44
aa06	646	7292	504	21,98	5902	19,06	199	5,03
kl01	55	7479	47	14,54	5915	20,91	1192	1,19
kl02	71	36699	69	2,82	16542	54,93	0	0,10
nw01	135	51975	135	0,00	49903	3,99	630	1,37
nw02	145	87879	145	0,00	85256	2,98	993	2,25

Continua na próxima página

Tabela 4.1 – Instâncias de Hoffman e Padberg [32]

instância	linhas	colunas	linhas PP	% linhas	colunas PP	% colunas	pClique	t(s)
nw03	59	43749	59	0,00	38964	10,94	0	0,10
nw04	36	87482	36	0,00	46190	47,20	0	0,35
nw05	71	288507	71	0,00	202603	29,77	0	2,97
nw06	50	6774	50	0,00	5977	11,76	0	0,08
nw07	36	5172	36	0,00	2064	39,91	0	0,07
nw08	24	434	24	0,00	356	17,97	0	0,00
nw09	40	3103	40	0,00	2305	25,72	0	0,00
nw10	24	853	24	0,00	659	22,74	0	0,00
nw11	39	8820	39	0,00	6488	26,44	0	0,01
nw12	27	626	27	0,00	454	27,48	0	0,00
nw13	51	16043	51	0,00	10905	32,03	0	0,02
nw14	73	123409	73	0,00	95178	22,87	0	0,65
nw15	31	467	31	0,00	407	12,85	145	0,02
nw16	139	148633	139	0,00	138951	6,51	0	0,73
nw17	61	118607	61	0,00	78186	34,08	0	0,49
nw18	124	10757	124	0,00	8458	21,37	702	3,52
nw19	40	2879	40	0,00	2145	25,49	0	0,00
nw20	22	685	22	0,00	536	21,75	33	0,00
nw21	25	577	25	0,00	421	27,04	117	0,02
nw22	23	619	23	0,00	521	15,83	47	0,00
nw23	19	711	19	0,00	424	40,36	176	0,02
nw24	19	1366	19	0,00	926	32,21	0	0,00
nw25	20	1217	20	0,00	844	30,65	0	0,00
nw26	23	771	21	8,70	468	39,30	205	0,04
nw27	22	1355	22	0,00	817	39,70	97	0,02
nw28	18	1210	18	0,00	582	51,90	189	0,04
nw29	18	2540	18	0,00	506	19,92	0	0,00
nw30	26	2653	26	0,00	1877	29,25	466	0,10
nw31	26	2662	26	0,00	1728	35,09	105	0,02
nw32	19	294	19	0,00	252	14,28	0	0,00
nw33	23	3068	23	0,00	2308	24,77	403	0,06
nw34	20	899	20	0,00	718	20,13	166	0,03
nw35	23	1709	23	0,00	1191	30,31	252	0,06
nw36	20	1783	20	0,00	1244	30,23	544	0,12
nw37	19	770	19	0,00	639	17,01	0	0,00
nw38	23	1220	23	0,00	726	40,49	382	0,15
nw39	25	677	25	0,00	565	16,54	88	0,01
nw40	19	404	19	0,00	336	16,83	0	0,00
nw41	17	197	17	0,00	177	10,15	0	0,00
nw42	23	1079	23	0,00	795	26,32	220	0,05
nw43	18	1072	18	0,00	983	8,30	0	0,00

Continua na próxima página

Tabela 4.1 – Instâncias de Hoffman e Padberg [32]

instância	linhas	colunas	linhas PP	% linhas	colunas PP	% colunas	pClique	t(s)
us02	100	13635	45	55,00	5766	57,71	1200	3,03
us03	77	85552	50	35,06	20632	75,88	2736	31,41
us04	163	28016	101	38,04	4209	84,98	187	2,35

Em [12] e [35] há uma listagem de métodos de pré-processamento. Optamos por utilizar somente os três procedimentos de redução apresentados porque eles se mostraram eficientes quanto à relação custo/benefício, ou seja, foram capazes de reduzir quantidades significativas de linhas e de colunas do PPC com baixo custo computacional.

4.3 MIP Cplex

Nesta seção apresentamos os resultados alcançados pela aplicação do software *Ilog Cplex 10.1* [1], para resolver as instâncias do PPC. Utilizamos *Cplex Callable Library* para obter a solução das instâncias considerando o modelo de programação inteira (2.1–2.2).

A tabela 4.2 mostra informações obtidas durante a execução da rotina *CPXmipopt* pelo Ilog Cplex, para solucionar as instâncias originais do PPC, não pré-processadas. Todas as soluções obtidas são ótimas.

Tabela 4.2 – Resultados da rotina *CPXmipopt* para instâncias do PPC

instância	custo ótimo	iter MIP	nNodes	nCortes	nDesigualdades	t(s)
aa01	56137	14211	128	26	613	37,08
aa02	30494	675	0	0	358	0,29
aa03	49649	1735	0	1	554	2,67
aa04	26374	19561	288	13	343	26,72
aa05	53839	2686	33	17	527	4,75
aa06	27040	1554	2	10	506	5,01
kl01	1086	182	3	10	47	0,89
kl02	219	381	19	2	69	4,11
nw01	114852	130	0	0	135	3,04
nw02	105444	157	0	0	145	6,00
nw03	24492	104	0	5	59	10,28
nw04	16862	3270	126	48	36	77,59
nw05	132878	127	0	0	71	15,84
nw06	7810	109	0	3	50	0,68
nw07	5476	22	0	0	36	0,14
nw08	35894	28	0	0	24	0,02
nw09	67760	48	0	0	40	0,08
nw10	68271	27	0	0	24	0,02

Continua na próxima página

Tabela 4.2 – Resultados da rotina *CPXmipopt* para instâncias do PPC

instância	custo ótimo	iter MIP	nNodes	nCortes	nDesigualdades	t(s)
nw11	116256	66	0	2	39	0,69
nw12	14118	23	0	0	27	0,02
nw13	50146	95	0	3	51	1,52
nw14	61844	156	0	0	73	6,70
nw15	67743	16	0	0	29	0,02
nw16	1181590	470	0	0	139	22,36
nw17	11115	212	2	20	61	71,44
nw18	340160	396	0	10	122	1,11
nw19	10898	44	0	0	40	0,10
nw20	16812	42	0	8	22	0,03
nw21	7408	16	0	2	25	0,03
nw22	6984	24	0	2	23	0,03
nw23	12534	39	0	0	18	0,02
nw24	6314	25	0	3	19	0,05
nw25	5960	25	0	2	20	0,04
nw26	6796	27	0	1	21	0,03
nw27	9933	17	0	1	22	0,05
nw28	8298	17	0	2	18	0,04
nw29	4274	80	1	12	18	0,20
nw30	3942	29	0	4	26	0,13
nw31	8038	33	0	4	26	0,10
nw32	14877	20	0	3	15	0,02
nw33	6678	21	0	0	23	0,11
nw34	10488	23	0	1	20	0,06
nw35	7216	19	0	0	23	0,06
nw36	7314	87	2	6	20	0,22
nw37	10068	20	0	1	19	0,03
nw38	5558	42	0	0	21	0,05
nw39	10080	15	0	2	25	0,05
nw40	10809	20	0	0	19	0,02
nw41	11307	14	0	1	17	0,02
nw42	7656	29	0	1	22	0,04
nw43	8904	29	0	1	17	0,04
us02	5965	95	0	0	45	0,69
us03	5338	81	0	0	50	5,04
us04	17854	192	0	0	98	1,16

Ao analisar os resultados obtidos na tabela 4.2, concluímos que o Cplex foi capaz de resolver todas as instâncias do PPC provenientes da *OR-Library* com facilidade, através da adição de cortes e desigualdades de clique durante a otimização. Os resultados demonstram a eficiência do Cplex na resolução dos problemas, considerando

a formulação de programação inteira do PPC. O valor da solução ótima obtida é utilizado nas tabelas que seguem com os demais resultados computacionais para análise da qualidade dos limites inferior e superior alcançados pelas heurísticas.

4.4 Busca Tabu

A busca tabu (BT) tem como ponto de partida a melhor solução viável obtida pela aplicação da heurística gulosa construtiva e dos critérios de seleção de colunas sobre as instâncias do PPC. Essa meta-heurística visa melhorar a qualidade das soluções iniciais obtidas, no intuito de fornecer um limite superior que é utilizado como parâmetro para a heurística de otimização pelo subgradiente.

Na etapa da busca local do algoritmo 3.1, selecionamos a melhor solução s' com base na avaliação da vizinhança $\mathcal{V}(s)$ da solução corrente s, com relação ao valor da função objetivo, à lista tabu e ao critério de aspiração. Selecionamos aleatoriamente $p = 50\%$ dos $C(|s|, 2)$ pares de colunas que são utilizados na busca por trocas entre colunas de s e de $\bar{s} = \mathcal{N} \backslash s$. Após a coleta de todos os *movimentos* possíveis, a BT seleciona aquele que apresenta maior descréscimo no custo da solução corrente ou menor acréscimo, caso não haja trocas que proporcionem redução no custo da solução corrente s. Verificamos então se o melhor movimento obtido na busca local está na *lista tabu*. Caso esteja, avaliamos se ele satisfaz o critério de aspiração. Caso satisfaça ele é realizado mesmo estando na *lista tabu*. Se o melhor *movimento* selecionado não estiver na *lista tabu*, ele é realizado e, em seguida, inserido na *lista tabu*. Dessa forma, os *movimentos* promovem uma transição entre soluções viáveis para o PPC a cada iteração da BT.

Atribuímos ao tamanho $|T|$ da *lista tabu* o valor 10, determinado empiricamente após uma série de testes realizados sobre as instâncias da literatura. Os movimentos que estiverem há $|T|$ iterações na lista são retirados dela. Para cada instância da literatura, realizamos a busca tabu até que a quantidade de $BT_{max} = 50$ iterações sem melhoria na melhor solução conhecida s^* seja alcançada.

A tabela 4.3 informa os resultados coletados para as 54 instâncias da *OR-Library* pré-processadas, utilizando o vetor de custos $c \in \mathbb{R}^+$ das colunas da matriz $\mathcal{A}$ do PPC após a execução da heurística construtiva gulosa e da busca tabu. Na tabela informamos o valor da solução ótima das instâncias, obtido pelo *CPXmipopt* do Cplex, o valor da solução inicial obtido a partir dos algoritmos gulosos (SIG) e sua distância percentual em relação ao valor ótimo, o valor do limite superior alcançado pela busca tabu e sua distância percentual em relação à solução ótima. O número de iterações (iter BT) e tempo de execução total em segundos (t(s)) da busca tabu também são informados.

Tabela 4.3 – Resultados da busca tabu

instância	custo ótimo	SIG	% SIG	LS BT	% LS BT	iter BT	t(s)
aa01	56137	57528	2,48	57528	2,48	50	28,86
aa02	30494	30497	0,01	30497	0,01	50	10,16
aa03	49649	50747	2,21	50702	2,12	90	46,07
aa04	26374	29888	13,20	29634	12,24	98	33,58
aa05	53839	55160	2,45	55160	2,45	50	23,16
aa06	27040	27293	0,94	27293	0,94	50	20,28
kl01	1086	1130	4,05	1099	1,20	101	5,69
kl02	219	234	6,85	229	4,57	55	16,82
nw03	24492	34917	42,56	27177	10,96	134	69,43
nw04	16862	20698	22,75	17282	2,49	85	46,48
nw05	132878	176114	32,54	173256	30,39	80	663,96
nw06	7810	13582	73,91	7810	0,00	85	3,31
nw07	5476	31988	484,15	25862	372,28	176	2,83
nw08	35894	46556	29,70	44230	23,22	67	0,13
nw09	67760	83166	22,74	78880	16,41	71	1,61
nw10	68271	83121	21,75	80319	17,65	106	0,42
nw11	116256	149127	28,27	147261	26,67	61	4,68
nw12	14118	16684	18,18	14118	0,00	56	0,30
nw13	50146	55554	10,74	50490	0,69	133	18,85
nw14	61844	68996	11,56	62518	1,09	66	216,08
nw15	67743	67800	0,08	67743	0,00	55	0,13
nw16	1181590	1181590	0,00	1181590	0,00	50	1,30
nw17	11115	37938	241,32	13326	19,89	115	205,29
nw18	340160	455826	34,00	452478	33,02	57	18,11
nw19	10898	18088	65,98	11172	2,51	61	0,55
nw20	16812	18873	12,26	17157	2,05	154	0,29
nw21	7408	9734	31,40	7408	0,00	60	0,30
nw22	6984	7610	8,96	6984	0,00	54	0,20
nw23	12534	15144	20,82	12534	0,00	65	0,15
nw24	6314	6424	1,74	6314	0,00	55	0,22
nw25	5960	9152	53,56	5960	0,00	52	0,08
nw26	6796	8432	24,07	6796	0,00	58	0,56
nw27	9933	15834	59,41	9933	0,00	56	0,30
nw28	8298	8298	0,00	8298	0,00	50	0,25
nw29	4274	4702	10,01	4274	0,00	54	0,40
nw30	3942	7226	83,31	4108	4,21	59	0,51
nw31	8038	9992	24,31	8038	0,00	57	0,32
nw32	14877	15609	4,92	14877	0,00	54	0,25
nw33	6678	9892	48,13	6678	0,00	51	0,26
nw34	10488	13140	25,29	10488	0,00	58	0,40

Continua na próxima página

Tabela 4.3 – Resultados da busca tabu

instância	custo ótimo	SIG	% SIG	LS BT	% LS BT	iter BT	t(s)
nw35	7216	7896	9,42	7216	0,00	60	0,53
nw37	10068	13503	34,12	10068	0,00	52	0,18
nw38	5558	6598	18,71	5558	0,00	52	0,22
nw39	10080	10758	6,73	10080	0,00	51	0,30
nw40	10809	12372	14,46	10809	0,00	60	0,39
nw41	11307	12474	10,32	11307	0,00	52	0,17
nw42	7656	10530	37,54	7656	0,00	54	0,34
nw43	8904	12370	38,93	8904	0,00	66	0,16

Os resultados encontrados mostram que a aplicação da meta-heurística busca tabu melhorou o custo da solução viável inicial em 42 das 54 instâncias. Para as instâncias *aa01*, *aa02*, *aa05*, *aa06* a busca tabu não foi capaz de melhorar o custo da solução viável encontrada pela heurística construtiva gulosa. No entanto, para as instâncias *nw16* e *nw28* a heurística gulosa encontrou uma solução com custo ótimo e, como conseqüência, a busca tabu não foi aplicada sobre elas. Em 46,30% das instâncias, uma solução ótima foi alcançada, com custo computacional razoável. No entanto, para as instâncias *nw01*, *nw02*, *nw36*, *us02*, *us03* e *us04* não obtivemos solução viável pela aplicação das heurística gulosa construtiva e, conseqüentemente, a busca tabu não foi aplicada a elas. Para essas instâncias, estimamos um limite superior, apresentado na tabela 4.4, que é dado pela soma dos custos das m' colunas de maior custo, onde $m' \leq m$ é o número de linhas da instância do PPC pré-processada. Com isso, todas as instâncias ficam aptas a serem submetidas às heurísticas lagrangeanas, cujos resultados computacionais são apresentados a seguir.

Tabela 4.4 – Estimativa de limite superior

instância	custo ótimo	estimativa LS	% estimativa LS
nw01	114852	1622590	1312,77
nw02	105444	1744940	1554,85
nw36	7314	113474	1451,46
us02	5965	64375	979,21
us03	5338	78433	1369,33
us04	17854	137098	667,88

Em geral o tempo de execução da heurística gulosa construtiva e da busca tabu juntas, para as instâncias pré-processadas do PPC, é maior que o tempo de execução da rotina *CPXmipopt* do Cplex.

4.5 Heurísticas Lagrangeanas

O algoritmo 3.3, apresentado no capítulo anterior, descreve a integração entre as heurísticas lagrangeanas e o processo de otimização pelo subgradiente. A seguir especificamos os parâmetros utilizados, bem como os limites inferior e superior obtidos para as instâncias do PPC.

Atribuímos o valor 100 ao parâmetro α, para todas as instâncias, exceto para *nw36* e para *us04*, cujo parâmetro α recebeu os valores 745 e 500, respectivamente, definidos após uma série de testes que avaliaram a qualidade do limite inferior obtido. Quando o número de iterações sem melhoria no valor do melhor limite inferior conhecido atinge o valor α, o valor do parâmetro π é reduzido pela metade.

As tabelas 4.5 e 4.6 informam o valor do custo de uma solução ótima, os valores dos limites superior e inferior obtidos e o GAP relativo entre esses valores, bem como o número de iterações executadas pela otimização pelo subgradiente e o tempo de execução em segundos das heurísticas lagrangeanas pura e de custo complementar, respectivamente, para as instâncias do PPC pré-processadas.

Em virtude da busca tabu ser executada em cada iteração da otimização pelo subgradiente em que há uma melhoria no melhor limite inferior corrente, resolvemos reduzir o parâmetro p da busca tabu, que indica a percentagem do total de pares de colunas da solução corrente analisado para a execução do procedimento de busca local, de 50% para 30% com o objetivo de reduzir o tempo de execução de cada chamada da busca tabu.

Tabela 4.5 – Resultados Heurística Lagrangeana Pura (HLP)

instância	custo ótimo	LS HL	% LS HL	LI HL	% LI HL	nIOS	GAP	t(s)
aa01	56137	56989	1,52	55504	1,13	6876	2,68	86,28
aa02	30494	30494	0,00	30494	0,00	470	0,00	20,91
aa03	49649	49649	0,00	49615	0,07	6680	0,07	83,53
aa04	26374	28898	9,45	25863	2,04	4939	11,73	61,31
aa05	53839	53935	0,18	53721	0,22	9439	0,40	232,08
aa06	27040	27112	0,27	26974	0,24	3764	0,51	353,07
kl01	1086	1086	0,00	1084	0,18	2182	0,18	50,34
kl02	219	220	0,46	216	1,37	2409	1,85	57,00
nw01	114852	114852	0,00	114852	0,00	826	0,00	1254,75
nw02	105444	105444	0,00	105444	0,00	46400	0,00	3062,82
nw03	24492	24492	0,00	24447	0,18	1620	0,18	105,33
nw04	16862	16862	0,00	16311	3,27	2335	3,38	385,85
nw05	132878	149136	12,24	132868	0,01	2335	12,24	3561,29
nw06	7810	7810	0,00	7640	2,18	1731	2,23	2,01
nw07	5476	5476	0,00	5476	0,00	1782	0,00	7,97

Continua na próxima página

Tabela 4.5 – Resultados Heurística Lagrangeana Pura (HLP)

instância	custo ótimo	LS HL	% LS HL	LI HL	% LI HL	nIOS	GAP	t(s)
nw08	35894	35894	0,00	35894	0,00	210	0,00	0,59
nw09	67760	67760	0,00	67760	0,00	298	0,00	5,64
nw10	68271	68271	0,00	68271	0,00	380	0,00	1,37
nw11	116256	116256	0,00	116255	0,00	1485	0,00	33,74
nw12	14118	14118	0,00	14118	0,00	44	0,00	0,01
nw13	50146	50146	0,00	50132	0,03	2241	0,03	47,54
nw14	61844	61844	0,00	61822	0,04	4591	0,04	3028,86
nw15	67743	67743	0,00	67743	0,00	289	0,00	0,03
nw16	1181590	1181590	0,00	1181590	0,00	513	0,00	93,41
nw17	11115	11196	0,73	10873	2,18	2655	2,97	875,38
nw18	340160	368964	8,47	338621	0,45	2101	8,96	219,35
nw19	10898	10898	0,00	10898	0,00	74	0,00	0,42
nw20	16812	16812	0,00	16626	1,11	1930	1,12	1,21
nw21	7408	7408	0,00	7380	0,38	1465	0,38	0,15
nw22	6984	6984	0,00	6942	0,60	1152	0,61	0,11
nw23	12534	12534	0,00	12317	1,73	1190	1,76	0,11
nw24	6314	6314	0,00	5842	7,48	1023	8,08	0,16
nw25	5960	5960	0,00	5852	1,82	1020	1,85	0,13
nw26	6796	6796	0,00	6743	0,78	1162	0,79	0,15
nw27	9933	9933	0,00	9878	0,55	1032	0,56	0,14
nw28	8298	8298	0,00	8169	1,55	1059	1,58	0,12
nw29	4274	4274	0,00	4170	2,43	2703	2,49	0,61
nw30	3942	3942	0,00	3727	5,45	2027	10,22	0,86
nw31	8038	8038	0,00	7980	0,72	1488	0,73	0,65
nw32	14877	14877	0,00	14570	2,06	1797	2,11	0,17
nw33	6678	6678	0,00	6484	2,91	1011	2,99	0,32
nw34	10488	10488	0,00	10454	0,32	1073	0,33	0,10
nw35	7216	7216	0,00	7206	0,14	1061	0,14	0,22
nw36	7314	7314	0,00	7260	0,74	27925	0,74	12,86
nw37	10068	10068	0,00	9962	1,05	998	1,06	0,07
nw38	5558	5558	0,00	5552	0,11	1557	0,11	0,22
nw39	10080	10080	0,00	9869	2,09	1202	2,09	0,11
nw40	10809	10809	0,00	10659	1,39	1422	1,39	0,14
nw41	11307	11307	0,00	10972	2,96	1007	3,05	0,03
nw42	7656	7656	0,00	7485	2,23	1226	2,28	0,16
nw43	8904	8904	0,00	8897	0,08	1150	0,08	0,13
us02	5965	5965	0,00	5965	0,00	165	0,00	9,26
us03	5338	5338	0,00	5329	0,17	76444	0,17	3995,30
us04	17854	17854	0,00	17732	0,68	46378	0,69	1340,71

Tabela 4.6 – Resultados Heurística Lagrangeana de Custo Complementar (HLCC)

instância	custo ótimo	LS HL	% LS HL	LI HL	% LI HL	nIOS	GAP	t(s)
aa01	56137	57528	2,48	55427	1,27	4127	3,79	94,43
aa02	30494	30494	0,00	30494	0,00	447	0,00	20,74
aa03	49649	50702	2,12	49605	0,09	2595	2,21	45,61
aa04	26374	29769	12,75	25856	2,07	2658	15,13	33,32
aa05	53839	53941	0,19	53722	0,22	4854	0,40	140,08
aa06	27040	27293	0,94	26974	0,24	3075	1,18	126,74
kl01	1086	1086	0,00	1084	0,19	2024	1,18	66,00
kl02	219	219	0,00	216	1,37	2356	1,38	64,07
nw01	114852	114852	0,00	114852	0,00	734	0,00	1079,85
nw02	105444	105444	0,00	105444	0,00	46368	0,00	3840,42
nw03	24492	25464	3,97	24446	0,19	1538	4,16	298,31
nw04	16862	16862	0,00	16311	3,27	2562	3,37	457,90
nw05	132878	149132	12,23	132873	0,00	32853	12,23	3711,21
nw06	7810	7810	0,00	7640	2,18	1731	2,22	1,93
nw07	5476	5476	0,00	5476	0,00	1853	0,00	4,26
nw08	35894	35894	0,00	35894	0,00	55	0,00	0,12
nw09	67760	67760	0,00	67760	0,00	263	0,00	5,15
nw10	68271	68271	0,00	68271	0,00	125	0,00	1,63
nw11	116256	116256	0,00	116255	0,00	2031	0,00	34,67
nw12	14118	14118	0,00	14118	0,00	44	0,00	0,07
nw13	50146	50322	0,35	50132	0,03	1987	0,37	86,56
nw14	61844	62514	1,08	61829	0,02	3093	1,10	2737,34
nw15	67743	67743	0,00	67743	0,00	289	0,00	0,07
nw16	1181590	1181590	0,00	1181590	0,00	513	0,00	94,83
nw17	11115	11142	0,24	10871	2,20	3049	2,49	875,17
nw18	340160	415154	22,05	338564	0,47	19559	22,62	235,95
nw19	10898	10898	0,00	10898	0,00	51	0,00	1,90
nw20	16812	17157	2,05	16626	1,11	2236	3,19	1,30
nw21	7408	7408	0,00	7380	0,38	1465	0,37	0,20
nw22	6984	6984	0,00	6942	0,60	1152	0,60	0,17
nw23	12534	12534	0,00	12317	1,73	1190	1,16	0,14
nw24	6314	6314	0,00	5842	7,48	1023	8,07	0,20
nw25	5960	5960	0,00	5852	1,82	1020	1,84	0,15
nw26	6796	6796	0,00	6743	0,78	1143	0,78	0,24
nw27	9933	9933	0,00	9878	0,55	1032	0,55	0,19
nw28	8298	8298	0,00	8169	1,56	1059	1,57	0,18
nw29	4274	4274	0,00	4170	2,43	2703	2,49	0,72
nw30	3942	3942	0,00	3727	5,47	1778	5,76	0,94
nw31	8038	8038	0,00	7980	0,72	1613	0,72	0,94
nw32	14877	14877	0,00	14570	2,07	1890	2,10	0,29

Continua na próxima página

Tabela 4.6 – Resultados Heurística Lagrangeana de Custo Complementar (HLCC)

instância	custo ótimo	LS HL	% LS HL	LI HL	% LI HL	nIOS	GAP	t(s)
nw33	6678	6678	0,00	6484	2,91	1011	2,99	0,37
nw34	10488	10488	0,00	10454	0,32	1073	0,32	0,23
nw35	7216	7216	0,00	7206	0,14	1061	0,13	0,22
nw36	7314	7314	0,00	7260	0,74	24461	0,74	12,58
nw37	10068	10068	0,00	9962	1,05	998	1,06	0,16
nw38	5558	5558	0,00	5552	0,11	1557	0,10	0,31
nw39	10080	10080	0,00	9869	2,09	1202	2,13	0,14
nw40	10809	10809	0,00	10659	1,39	1574	1,40	0,20
nw41	11307	11307	0,00	10972	2,96	1007	3,05	0,03
nw42	7656	7656	0,00	7485	2,24	1226	2,28	0,23
nw43	8904	8904	0,00	8897	0,08	1150	0,07	0,16
us02	5965	5965	0,00	5965	0,00	374	0,00	56,14
us03	5338	5338	0,00	5331	0,13	68027	0,13	3671,18
us04	17854	17854	0,00	17732	0,68	45065	0,68	1362,62

Um comparativo entre os resultados alcançados para as 54 instâncias da *OR-Library* mostra que as heurísticas HLP e HLCC alcançaram o mesmo limite superior em 41 instâncias. HLP obteve melhor limite superior nas instâncias *aa01*, *aa03*, *aa04*, *aa05*, *aa06*, *nw03*, *nw13*, *nw14*, *nw18* e *nw20*, enquanto HLCC obteve melhor limite superior em *kl02*, *nw05* e *nw17*. Em 46 instâncias, HLP atingiu a solução ótima, enquanto HLCC obteve a solução ótima em 42 instâncias. Quanto ao limite inferior, em 45 instâncias, HLP e HLCC alcançaram o mesmo resultado, sendo que em 13 instâncias foi atingido o custo da solução ótima. Para as instâncias *aa04*, *nw03*, *nw17* e *nw18*, HLP obteve melhor limite inferior, enquanto para as instâncias *aa03*, *aa05*, *nw05*, *nw14* e *us03*, HLCC obteve melhor limite inferior. Os resultados mostram que tanto a HLP quanto a HLCC alcançaram a solução ótima com prova de otimalidade em 13 das 54 instância do PPC. Em média, a HLCC apresentou menor tempo de execução que a HLP.

4.6 *Branch and Bound*

Nesta seção apresentamos os resultados alcançados pelo nosso algoritmo de B&B que utiliza o software Cplex [1] para resolver a relaxação linear das instâncias do PPC pré-processadas, através do método *dual simplex*.

Na tabela 4.7, *LS Inic* indica o limite superior dado como entrada para o B&B, que corresponde ao melhor limite superior alcançado pelas heurísticas lagrangeanas. *LI Root* indica o valor da solução da relaxação linear do nó raiz da árvore de B&B. A coluna *sol BB* indica que nosso B&B alcançou a solução ótima em todas as instâncias do PPC

(GAP = 0), com exceção de *aa01*, *aa04* e *nw04*. Nosso B&B não foi capaz de resolver essas três instâncias, devido a limitações de recursos de memória no *cluster* onde o processo foi executado. A dificuldade apresentada pelo B&B para resolver as instâncias *aa01–06* se deve ao número relativamente grande de linhas por elas apresentado em comparação com as demais instâncias.

Na tabela 4.7 também informamos o custo de uma solução ótima, o número de nós da árvore de B&B que foram gerados, explorados e podados durante a otimização, bem como o tempo de execução em segundos. Para as instâncias *aa02*, *nw01*, *nw02*, *nw05*, *nw07*, *nw08*, *nw09*, *nw10*, *n12*, *nw14*, *nw15*, *nw16*, *nw19*, a resolução da relaxação linear do nó raiz da árvores de B&B resultou numa solução inteira com valor igual a da solução ótima. Para as demais instâncias, os resultados mostram que o B&B teve facilidade em resolvê-las, devido ao pequeno número de nós que foram gerados até que uma solução ótima fosse encontrada.

Os resultados alcançados pelo B&B provaram a otimalidade do limite superior obtido pelas heurísticas lagrangeanas para as instâncias do PPC em que o limite superior obtido por elas é igual ao valor da solução ótima, o que demonstra a qualidade dos limites alcançados para a resolução de instâncias da literatura.

Tabela 4.7 – Resultados *Branch and Bound*

instância	custo ótimo	LS Inic	LI Root	sol BB	gerados	expl	podados	t(s)
aa02	30494	30494	30494	30494	0	0	0	0,243
aa03	49649	50702	49616,4	49649	20	10	10	4,596
aa05	53839	53941	53735,9	53839	116	58	58	23,338
aa06	27040	27293	26977,2	27040	96	48	48	24,407
kl01	1086	1086	1084	1086	74	37	22	2,902
kl02	219	219	215,25	219	1694	847	285	279,140
nw01	114852	114852	114852	114852	0	0	0	2,181
nw02	105444	105444	105444	105444	0	0	0	3,936
nw03	24492	24492	24492	24492	2	1	2	2,648
nw05	132878	149132	149132	132878	0	0	0	10,273
nw06	7810	7810	7810	7810	6	3	3	0,308
nw07	5476	5476	5476	5476	0	0	0	0,131
nw08	35894	35894	35894	35894	0	0	0	0,025
nw09	67760	67760	67760	67760	0	0	0	0,219
nw10	68271	68271	68271	68271	0	0	0	0,020
nw11	116256	116256	116256	116256	2	1	2	0,432
nw12	14118	14118	14118	14118	0	0	0	0,022
nw13	50146	50146	50146	50146	2	1	0	0,835
nw14	61844	61844	61844	61844	0	1	0	4,867
nw15	67743	67743	67743	67743	0	1	0	0,030

Continua na próxima página

Tabela 4.7 – Resultados *Branch and Bound*

instância	custo ótimo	LS Inic	LI Root	sol BB	gerados	expl	podados	t(s)
nw16	1181590	1181590	1181590	1181590	0	1	0	16,123
nw17	11115	11142	10875,8	11115	20	10	10	19,638
nw18	340160	368964	338864	340160	2	1	0	0,904
nw19	10898	10898	10898	10898	0	1	0	0,159
nw20	16812	16812	16626	16812	4	2	3	0,127
nw21	7408	7408	7380	7408	2	1	0	0,032
nw22	6984	6984	6942	6984	2	1	0	0,064
nw23	12534	12534	12317	12534	16	8	7	0,100
nw24	6314	6314	5843	6314	6	3	3	0,175
nw25	5960	5960	5852	5960	4	2	2	0,122
nw26	6796	6796	6743	6796	4	2	2	0,059
nw27	9933	9933	9877,5	9933	2	1	0	0,087
nw28	8298	8298	8169	8298	4	2	2	0,142
nw29	4274	4274	4185,33	4274	12	6	6	0,240
nw30	3942	3942	3726,8	3942	4	2	0	0,218
nw31	8038	8038	8022	8038	6	3	3	0,299
nw32	14877	14877	14570	14877	8	4	4	0,039
nw33	6678	6678	6484	6678	2	1	0	0,218
nw34	10488	10488	10453,5	10488	2	1	0	0,026
nw35	7216	7216	7206	7216	2	1	0	0,095
nw36	7314	7314	7260	7314	28	14	13	0,387
nw37	10068	10068	9961,5	10068	2	1	0	0,023
nw38	5558	5558	5552	5558	2	1	0	0,102
nw39	10080	10080	9868,5	10080	6	3	3	0,060
nw40	10809	10809	10658,2	10809	4	2	2	0,032
nw41	11307	11307	10972,5	11307	2	1	0	0,030
nw42	7656	7656	7485	7656	6	3	3	0,175
nw43	8904	8904	8897	8904	2	1	0	0,085
us02	5965	5965	5965	5965	0	0	0	0,645
us03	5338	5338	5338	5338	0	0	0	2,192
us04	17854	17854	17731,7	17854	6	3	3	1,502

4.7 Resultados da Literatura

No intuito de avaliar a performance das heurísticas apresentadas, comparamos nossos resultados com os de algoritmos existentes na literatura que utilizam as instâncias de Hoffman e Padberg [9] para o PPC.

4.7.1 Algoritmo genético

Na tabela 4.8 apresentamos os melhores resultados obtidos pela heurística baseada em algoritmos genéticos (AG) desenvolvida por Chu e Beasley [15]. Essa heurística

caracteriza-se por apresentar componentes tais como funções *fitness* e *unfitness* separadas, esquema de mutação adaptativa, seleção de emparelhamento (*matching*), função de substituição da população por *ranking* e um operador de melhoramento heurístico. Trata-se de uma das melhores heurísticas da literatura para o PPC.

Em *AG* estão os resultados obtidos em uma estação de trabalho Silicon Graphics Indico (R4000, 100 MHz). Em *% AG* mostramos a diferença relativa entre o limite superior obtido pelo AG e o custo da solução ótima que obtivemos na seção 4.3. Em *t(s)* indicamos o tempo médio em segundos que o AG levou para alcançar sua melhor solução. O AG não foi capaz de encontrar solução viável para as instâncias *aa01*, *aa03*, *aa05*, e *nw01*.

Tabela 4.8 – Resultados do AG de Chu e Beasley [15]

instância	custo ótimo	AG	% AG	t(s)
aa02	30494	30500	0,02	1145,4
aa04	26374	28261	7,04	1731,7
aa06	27040	27883	3,12	2114,8
kl01	1086	1086	0,00	159,6
kl02	219	219	0,00	485,7
nw02	105444	108816	3,20	15132,0
nw03	24492	24492	0,00	2782,0
nw04	16862	16862	0,00	3458,6
nw05	132878	134170	0,97	24914,5
nw06	7810	7810	0,00	326,4
nw07	5476	5476	0,00	47,6
nw08	35894	35894	0,00	0,7
nw09	67760	67760	0,00	14,2
nw10	68271	68271	0,00	1,8
nw11	116256	116256	0,00	69,8
nw12	14118	14118	0,00	3,5
nw13	50146	50146	0,00	389,4
nw14	61844	62262	0,68	9782,0
nw15	67743	67743	0,00	1,4
nw16	1181590	1181590	0,0	11810,5
nw17	11115	11115	0,00	5650,7
nw18	340160	345130	1,46	1540,8
nw19	10898	10898	0,00	69,7
nw20	16812	16812	0,00	3,2
nw21	7408	7408	0,00	1,1
nw22	6984	6984	0,00	0,5
nw23	12534	12534	0,00	1,5
nw24	6314	6314	0,00	4,9
nw25	5960	5960	0,00	3,6

Continua na próxima página

Tabela 4.8 – Resultados do AG de Chu e Beasley [15]

instância	custo ótimo	AG	% AG	t(s)
nw26	6796	6796	0,00	4,4
nw27	9933	9933	0,00	2,6
nw28	8298	8298	0,00	3,0
nw29	4274	4274	0,00	48,8
nw30	3942	3942	0,00	28,1
nw31	8038	8038	0,00	42,2
nw32	14877	14877	0,00	1,9
nw33	6678	6678	0,00	5,5
nw34	10488	10488	0,00	2,2
nw35	7216	7216	0,00	4,5
nw36	7314	7314	0,00	46,2
nw37	10068	10068	0,00	2,3
nw38	5558	5558	0,00	6,7
nw39	10080	10080	0,00	1,1
nw40	10809	10809	0,00	1,4
nw41	11307	11307	0,00	0,9
nw42	7656	7656	0,00	16,3
nw43	8904	8904	0,00	6,5
us02	5965	5965	0,00	76,8
us03	5338	5338	0,00	1350,2
us04	17854	17854	0,00	135,1

Nossa HLP alcançou solução viável para as instâncias *aa01* e *aa05* com distância relativa para a solução ótima de 1,52% e 0,18% respectivamente. Para as instâncias *aa03* e *nw01*, a HLP alcançou a solução ótima, para as quais a prova de otimalidade é mostrada pelo resultado do nosso B&B. Em 42 instâncias do PPC, o melhor limite superior alcançado pelas heurísticas lagrangeanas e pelo AG foi o mesmo. A HLP obteve melhor resultado nas instâncias *aa01*, *aa02*, *aa03*, *aa05*, *aa06*, *nw01*, *nw02*, *nw14* do que o AG, enquanto a HLCC foi melhor que o AG nas instâncias *aa01*, *aa02*, *nw01* e *nw02*. Somente para as instâncias *aa04*, *nw05*, *nw17* e *nw18* o AG obteve melhor resultado do que a HLP e a HLCC.

Em geral o tempo de execução das heurísticas lagrangeanas é inferior ao do AG. No entanto, deve ser levada em consideração a diferença entre as configurações das máquinas utilizadas para a execução das nossas heurísticas e do AG ao analisar a performance dos algoritmos em relação ao tempo de execução.

4.7.2 Algoritmo função dual subaditiva

Klabjan [34] apresenta um algoritmo prático que computa uma função dual subaditiva ótima (FSO) para problemas de programação inteira. Trata-se do primeiro trabalho que

reporta experimentos computacionais decorrentes da computação de uma FSO. Em [35] o autor mostra que o algoritmo prático é tratável para obter uma FSO para problemas de pequeno e médio portes do PPC, utilizando pré-processamento, desigualdades de clique, heurística de expansão baseada em programação linear, procedimento de geração de linhas e um esquema de *Branch-and-Cut*. Em [35] o autor também mostra que com o uso de funções subaditivas apropriadas é possível realisar análise de sensibilidade que fornece melhores resultados do que a análise que utiliza vetores duais de programação linear.

A tabela 4.9 de [35] informa os resultados alcançados pelos experimentos computacionais que foram executados em uma máquina IBM Thinkpad 570 com processador Pentium III de 333 MHz e 196 MB de memória, para resolver as instâncias do PPC de Hoffman e Padberg [32]. O autor utilizou o Visual Studio C++ versão 6.0 e o *software* Cplex versão 6.5 para resolver problemas de programação linear.

Tabela 4.9 – Resultados de Klabjan [35]

	Tamanho		Pré-processamento			Tempo				CPLEX
instância	linhas	colunas	linhas	colunas	t(s)	estágio 1	estágio 2	|E|	|N\S|	t(s)
aa01	823	8904	605	7531	32	3803	?	?	95	723
aa03	825	8627	537	6695	30	25	171	142	0	88
aa04	426	7195	342	6123	17	?	?	?	?	908
aa05	801	8308	521	6236	21	106	?	?	1	97
aa06	646	7292	486	5858	23	52	?	?	5	39
kl01	55	7479	47	5915	49	12	336	100	13	14
kl02	71	36699	69	16542	8	240	?	?	60	118
nw03	59	43749	53	38958	6	29	0	19	0	48
nw04	36	87482	35	46189	16	5757	?	?	998	378
nw06	50	6774	37	5964	45	10	0	58	0	8
nw11	39	8820	28	6436	28	4	0	3	0	6
nw13	51	16043	48	10900	4	9	0	4	0	18
nw18	124	10757	81	7861	90	100	0	292	0	13
nw20	22	685	22	536	0	0	0	13	0	0
nw21	25	577	25	421	0	0	0	4	0	0
nw22	23	619	23	521	0	0	0	7	0	0
nw23	19	711	15	416	0	4	0	52	0	2
nw24	19	1366	19	926	2	1	0	52	0	1
nw25	20	1217	20	844	0	0	0	9	0	0
nw26	23	771	21	468	1	0	0	7	0	1
nw27	22	1355	22	817	3	0	0	4	0	0
nw28	18	1210	18	582	3	0	0	10	0	0
nw29	18	2540	18	2034	6	3	19	50	4	1
nw30	26	2653	26	1877	22	1	0	27	0	1

Continua na próxima página

Tabela 4.9 – Resultados de Klabjan [35]

	Tamanho		Pré-processamento			Tempo				CPLEX
instância	linhas	colunas	linhas	colunas	t(s)	estágio 1	estágio 2	|E|	|N\S|	t(s)
nw31	26	2662	26	1728	14	1	0	13	0	1
nw32	19	294	17	250	0	2	0	28	0	1
nw33	23	3068	23	2308	26	2	5	16	1	1
nw34	20	899	20	718	2	0	0	3	0	1
nw35	23	1709	23	1191	7	0	0	3	0	1
nw36	20	1783	20	1244	15	5	14	67	12	0
nw37	19	770	19	639	0	0	0	3	0	0
nw38	23	1220	20	722	9	0	0	6	0	1
nw39	25	677	25	565	1	0	0	9	0	0
nw40	19	404	19	336	0	0	0	11	0	0
nw41	17	197	17	177	0	0	0	4	0	0
nw42	23	1079	20	791	4	1	0	30	0	0
nw43	18	1072	17	982	0	0	0	3	0	0
us02	100	13635	44	5197	289	247	0	287	0	17
us04	163	28016	95	4080	53	3	0	25	0	68

Segundo o autor, as instâncias do PPC, cuja resolução da relaxação linear apresenta solução inteira, foram omitidas na tabela 4.9. Devido à limitação de recursos de memória física, as instâncias *nw05*, *nw17* e *us01* não foram resolvidas. A instância *nw16* foi resolvida na fase de pré-processamento. Na tabela 4.9, mostramos o tamanho das instâncias antes e após a aplicação do pré-processamento. Em virtude do autor ter utilizado outros procedimentos de redução além dos que utilizamos, as quantidade de linhas e de colunas eliminadas nas instâncias do PPC foi igual ou pouco superior à quantidade que apresentamos na tabela 4.1. Isso mostra o quanto os procedimentos de redução que implementamos foram eficientes na eliminação de linhas e de colunas nas instâncias avaliadas.

O algoritmo apresentado por Klabjan [35] é composto de dois estágios, cujos tempos de execução em segundos estão presentes na tabela 4.9. O primeiro estágio é destinado a encontrar uma FSO e uma solução primal ótima, enquanto o segundo estágio recebe a FSO e computa uma FSO geradora. As colunas $|E|$ e $|N\backslash E|$ são cardinalidades de conjuntos utilizados para computar a FSO (veja [35] para mais informações). O símbolo "?" indica que os algoritmos não foram capazes de resolver as instâncias do PPC no tempo limite de duas horas. A última coluna da tabela mostra o tempo de execução de um *Branch and Cut* que utiliza Cplex para resolver as instâncias do PPC.

Segundo Klabjan, os algoritmos empregados não foram capazes de finalizar o estágio 1 no tempo limite estipulado, para resolver a instância *aa04*, enquanto para as instâncias *aa01*, *aa05*, *aa06*, *kl02* e *nw04* somente o estágio 1 foi resolvido.

Comparando os resultados do nosso B&B com o *Branch and Cut* de Klabjan [35], temos que nosso B&B foi capaz de encontrar a solução ótima para 51 das 54 instâncias do PPC num tempo de execução competitivo com o apresentado pelo algoritmo de [35].

Capítulo 5

Considerações Finais e Trabalhos Futuros

Neste trabalho avaliamos heurísticas lagrangeanas baseadas em algoritmos gulosos, meta-heurística busca tabu e método de otimização pelo subgradiente para resolver instâncias *benchmark* da literatura do Problema de Particionamento de Conjuntos (PPC). As heurísticas usam informações da solução do problema lagrangeano para modificar os custos das colunas e encontrar limites superiores e inferiores de boa qualidade. Empregamos um esquema de resolução exato do tipo *Branch and Bound* (B&B) para comprovar a qualidade dos resultados encontradas por nossas heurísticas. Utilizamos rotinas de pré-processamento iterativas para reduzir as dimensões das instâncias do PPC avaliadas, no intuito de melhorar a eficiência computacional das heurísticas empregadas.

Experimentos computacionais indicam que o pré-processamento promove redução de 5,91% das linhas e 27,35% das colunas, em média, nas instâncias da *OR-Library*. Constatamos que a rotina *CPXmipopt* da *Cplex Callable Library* resolve todas as instâncias do PPC analisadas com facilidade, através da aplicação de cortes e de desigualdades de clique. A aplicação da Heurística Lagrangeana Pura (HLP) leva à obtenção de uma solução ótima em 46 das 54 instâncias do PPC, ao passo que em 38 das 54 instâncias foi alcançado um limite inferior da ordem de 2% do custo da solução ótima. Constatamos também que a aplicação da nossa estratégia de B&B resultou na obtenção de uma solução ótima em 51 das 54 instâncias, sendo que em 13 delas a otimalidade foi alcançada pela resolução da relaxação linear do nó raiz da árvore de busca do B&B.

Os resultados obtidos pelas heurísticas lagrangeanas comparados com os da heurística baseada em algoritmo genético (AG) de Chul e Beasley [15] mostram que somente em 4 das 54 instâncias do PPC avaliadas o AG obteve melhores resultados e

que as heurísticas lagrangeanas são capazes de obter solução viável para as instâncias em que o AG não obteve solução.

Comparando os resultados das heurísticas lagrangeanas com os do algoritmo que computam uma função subaditiva ótima para o PPC, constatamos que o esquema de B&B que implementamos obteve uma solução ótima com tempo de execução inferior ao do *Branch and Cut*, presentes em Klabjan [35], para a maioria das instâncias avaliadas. Portanto os resultados obtidos fornecem uma forte evidência de que nossas heurísticas lagrangeanas são competitivas com as melhores heurísticas existentes na literatura para o PPC.

Como perspectivas de trabalhos futuros, pretendemos incorporar técnicas de planos de corte e desenvolver desigualdade válidas para o PPC, buscando melhorar os resultados obtidos pelas nossas heurísticas. Pretendemos também investigar como elas se comportam ao serem aplicadas na resolução de instâncias mais complexas desse problema.

Referências Bibliográficas

[1] Cplex-ilog, copyright ilog 1997–2006, version 10.1. 2006.

[2] A.O. Alves and R. Andrade. Heurísticas lagrangeanas para a otimização de particionamento de conjuntos. XXXIX Simpósio Brasileiro de Pesquisa Operacional, 2007.

[3] R. Andrade, A. Lucena, and N. Maculan. Using lagrangian dual information to generate degree constrained spanning trees. *Discrete Applied Mathematics*, 154:703–717, 2005.

[4] A. Atamtürk, G.L. Nemhauser, and M.W.P. Savelsbergh. A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics*, 1(2):247–259, 1995.

[5] E. Balas and A. Ho. Set covering algorithms using cutting planes, heuristics and subgradient optimization: a computational study. *Mathematical Programming*, 12:37–60, 1980.

[6] E. Balas and M.W. Padberg. Set partitioning: a survey. *SIAM Review*, 18:710–760, 1976.

[7] R. Baldacci, E. Hadjiconstantinou, V. Maniezzo, and A. Mingozzi. A new method for solving capacitated location problems based on a set partitioning approach. *Computers & Operations Research*, 29(4):365–386, 2002.

[8] F. Barahona and R. Anbil. On some difficult linear programs coming from set partitioning. *Discrete Applied Mathematics*, 118(1–2):3–11, 2002.

[9] J.E. Beasley. Or-library: Distributing test problems by electronic mail. *Operational Research Society*, 41:1069–1072, 1990.

[10] J.E. Beasley. *Lagrangean Relaxation*. The Management School, Imperial College, London SW7 2AZ, England, 1992.

[11] J.E. Beasley and B. Cao. A dynamic programming based algorithm for the crew scheduling problem. *Elsevier Science*, 25(7/8):567–582, 1998.

[12] R. Borndörfer. *Aspects of set packing, partitioning and covering.* Ph.d. thesis, Technical University of Berlin, Germany, 1998.

[13] J. Bramel and D. Simchi-Levi. *The Logic of Logistics: Theory, Algorithms and Applications for Logistics Management.* Springer-Verlag, New York, 1997.

[14] C. Bryun. *Lower Bounds for Large-Scale Set Partitioning Problems.* Ph.d. thesis, Technical University of Berlin, Germany, 2001.

[15] P.C. Chu and J.E. Beasley. Constraint handling in genetic algorithms: The set partitioning problem. *Journal of Heuristics*, 4(4):323–357, 1998.

[16] V. Chvàtal. A greedy heuristic for the set covering problem. *Mathmatics of Operations Research*, 4:233–235, 1979.

[17] A. Cobham, R. Fridshal, and J.H. North. An application of linear programming to the minimization of boolean functions. *The Journal of Symbolic Logic*, 30:247, 1965.

[18] G. Desaulniers, J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M. M. Solomon, and F. Soumis. Crew pairing at air france. *European Journal of Operational Research*, 97(2):245–259, 1997.

[19] T. Emden-Weinert and M. Proksch. Best practice simulated annealing for the airline crew scheduling problem. *Journal of Heuristics*, 5(4), 1999.

[20] A.T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004.

[21] M. Eso. *Parallel branch and cut for set partitioning.* Ph.d. thesis, Cornell University, 1999.

[22] M.L. Fisher. The lagrangian relaxation method of solving integer programming problems. *Management Science*, 27:1–18, 1981.

[23] M.L. Fisher and P. Kedia. Optimal solutions of set covering and partitioning problems using dual heuristics. *Management Science*, 36(6), 1990.

[24] H.A. Fleuren. *A computational study of set partitioning approach for vehicle routing and scheduling problems.* Ph.d. thesis, University of Twente, The Netherlands, 1988.

[25] M.R. Garey and D.S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[26] R.S. Garfinkel and G.L. Nemhauser. The set-partitioning problem: Set covering with equality constraints. *Operations Research*, 17:848–856, 1969.

[27] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical programming study*, 2:82–114, 1974.

[28] F. Glover. Future paths for integer programming and artificial intelligence. *Computers and Operations Research*, 13(5), 1986.

[29] F. Glover. Tabu search and adaptive memory programming – advances, applications and challenges. *Interfaces in Computer Science and Operations Research*, pages 1–75, 1996.

[30] M.C. Goldbarg and H.P.L. Luna. *Otimização Combinatória e Programação Linear – Modelos e algoritmos.* Editora Campus, 2ed edition, 2000.

[31] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. Congress on Numerical Methods in Combinatorial Optimization, 1986.

[32] K.L. Hoffman and M.W. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Science*, 39(6):657–682, 1993.

[33] B.L. Hulme and L.S. Baca. Set covering, partition and packing. *Mathematical programming*, 31:163–171, 1984.

[34] D. Klabjan. A new subadditive approach to integer programming. Technical report, University of Illinois at Urbana-Champaign, 2003.

[35] D. Klabjan. A practical algorithm for computing a subadditive dual function for set partitioning. *Computational Optimization and Applications*, 29:347–368, 2004.

[36] S. Lavoie, M. Minoux, and E. Odier. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35(1):45–58, 1988.

[37] R.E. Marsten. An algorithm for large set partitioning problems. *Management Science*, 20:774–787, 1974.

[38] W.M. Nawijn. Optimizing the performance of a blood analyser: Application of the set partitioning problem. *University of Twente Memorandum 626*, 1987.

[39] H.T. Ozdemir and C.K. Mohan. Flight graph based genetic algorithm for crew scheduling in airlines. *Information Sciences*, 133(3–4):165–173, 2001.

[40] A.L.G. Pimentel. *Uma abordagem heurística para a solução de problemas de recobrimento de conjuntos de grande porte, com aplicação na alocação de tripulações para companhias aéreas.* PhD thesis, Universidade Federal do Rio de Janeiro, 2005.

[41] C.C. Ribeiro, E. Uchoa, and R.F. Werneck. A hybrid grasp with perturbations for the steiner problem in graph. *INFORMS Journal on Computing*, 14:228–246, 2002.

[42] M.W.P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal of Computing*, 6:445–454, 1994.

[43] H.D. Sherali and Y.H. Lee. Tighter representations for set partitioning problems. *Discrete Applied Mathematics*, 68(1-2):153–167, 1996.

[44] M.G.C. van Krieken, H.A. Fleuren, and M.J.P. Peeters. *Solving Set Partitioning Problems using lagrangian relaxation.* PhD thesis, Tilburg University, 2005.

[45] F.J. Vasko and G.R. Wilson. An efficient heuristic for large set covering problems. *Naval Research Logistics Quarterly*, 31:163–171, 1984.