Universidade do Minho
Escola de Engenharia

Carlos Miguel Marques Peixoto

Modelos e Algoritmos para o Problema de Minimização de Padrões

Março de 2009

Carlos Miguel Marques Peixoto **Modelos e Algoritmos para o Problema de Minimização de Padrões**

UMinho | 2009

Universidade do Minho
Escola de Engenharia

Carlos Miguel Marques Peixoto

Modelos e Algoritmos para o Problema de Minimização de Padrões

Tese de Mestrado em Engenharia Industrial

Trabalho efectuado sob a orientação do
Professor Doutor Cláudio Manuel Martins Alves

Março de 2009

Resumo

Nesta dissertação, estudamos um problema de optimização combinatória designado por Problema de Minimização de Padrões. O problema é um problema de corte no qual se consideram custos de *setup*. Existe um número muito reduzido de métodos para a resolução exacta do Problema de Minimização de Padrões. Aqueles que são descritos na literatura apenas permitem resolver algumas instâncias de pequena dimensão. Com esta dissertação, pretendemos contribuir para a resolução exacta do problema explorando um modelo que foi proposto recentemente na literatura.

Os modelos de Programação Inteira propostos na literatura são descritos na primeira parte do texto. Descrevemos também os algoritmos exactos que foram definidos com base nesses modelos e apresentamos algumas das melhores heurísticas de resolução que foram desenvolvidas para este problema.

Na segunda parte da dissertação, analisamos em detalhe um modelo proposto recentemente na literatura para o Problema de Minimização de Padrões. Esse modelo é um modelo de Programação Inteira, obtido com base numa nova decomposição de um modelo não-linear. Exploramos formas de reforçar esse modelo, e analisamos algoritmos que permitam obter soluções óptimas inteiras a partir do método de partição e avaliação e do método de geração de colunas.

A diferença entre os algoritmos reside nas regras de partição que foram usadas. É sabido que combinar o método de partição e avaliação com o método de geração de colunas não é trivial. A partição feita com base nas variáveis do modelo de geração de colunas provoca a regeneração das colunas presentes no problema mestre. A forma de evitar esse problema passa por aumentar a complexidade do subproblema. Nesta dissertação, as regras de partição são baseadas em variáveis originais, e como tal não alteram significativamente a dificuldade dos subproblemas de geração de colunas.

Foram conduzidos experiências computacionais para avaliar a qualidade dos esquemas de partição propostos. Essas experiências foram realizadas usando instâncias da literatura, e sem recorrer a qualquer heurística. Os resultados das experiências são apresentados no final da dissertação.

Abstract

In this dissertation, we study a combinatorial optimization problem called the Pattern Minimization Problem. The problem is a cutting stock problem in which setup costs are considered. The number of exact algorithms that were proposed in the literature for this problem is very small. Those that were proposed can only solve a limited number of small and medium instances. With this dissertation, our aim is to contribute to the exact resolution of this problem by exploring a model that was proposed recently in the literature.

The Integer Programming models proposed in the literature are described in the first part of this text. We also describe the exact algorithms that were defined based on these models and we present the best heuristics that were developed to solve this problem.

In the second part of this dissertation, we analyse in detail a model proposed recently in the literature for the Pattern Minimization Problem. This model is an Integer Programming model that is obtained by applying a new decomposition to a non-linear model. We explore ways of strengthening the model, and we analyse algorithms for computing integer solution using the branch-and-bound method and the column generation method.

The difference among our algorithms is in the branching scheme that is used. It is well known that combinig branch-and-bound with column generation is not trivial. When the partition is done on the variables of the column generation model, regeneration occurs. To avoid this regeneration, we have to increase the complexity of the pricing subproblem. In this dissertation, the branching rules are based on original variables and hence the complexity of the subproblem is almost unchanged.

Computational experiments were conducted to evaluate the quality of the branching schemes proposed. These experiences were conducted on instances from the literature, and without using any heuristic. The results of these experiments are presented at the end of this dissertation.

Agradecimentos

Ao longo do tempo em que este trabalho teve lugar, houve sem dúvida muitas pessoas que de uma forma ou de outra foram importantes e me ajudaram nesta caminhada. Entre elas encontram-se colegas de curso, professores, amigos ou familiares. A todas elas, gostaria de exprimir a minha gratidão.

Entre todas essas pessoas, gostaria contudo de destacar o meu orientador, o Professor Cláudio Alves, a quem agradeço a orientação, a dedicação, a compreensão, a ajuda e a presença nos momentos mais difíceis. Sem a sua orientação, nunca teria sido possível chegar a este ponto.

Finalmente, quero também aqui deixar uma palavra de agradecimento ao Professor Valério de Carvalho.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os processos de corte e empacotamento surgem em diversas indústrias desde a metalurgia à indústria do papel passando pela indústria automóvel. Consistem em determinar a maneira mais eficiente de associar objectos de pequena dimensão a objectos de maior dimensão (processos de empacotamento), ou de modo equivalente, em determinar a forma mais eficiente de retirar ou cortar objectos pequenos a partir de objectos maiores (processos de corte). Esses problemas não ocorrem exclusivamente na indústria. Podem ocorrer também em diversos outros contextos como por exemplo na gestão das memórias de computadores, ou ainda em problemas de investimento (problemas de mochila).

Nesta dissertação, analisamos um problema auxiliar ligado ao problema de corte a uma dimensão: o problema de minimização de padrões. Trata-se de um problema complexo que tem sido sobretudo resolvido por procedimentos heurísticos. Neste trabalho, abordamos a resolução exacta do problema. As abordagens que desenvolvemos nos próximos capítulos baseiam-se essencialmente em métodos de Programação Inteira tais como o método de geração de colunas, o método de partição e avaliação sucessivas e o método dos planos de corte. Daqui em diante, e por motivos de clareza, referir-nos-emos quase em exclusivo aos problemas de corte em detrimento dos problemas de empacotamento.

Quando são cortados objectos pequenos (normalmente designados por itens) a partir de um objecto maior, surgem tipicamente dois problemas. O primeiro é ter de se escolher o tamanho adequado para o objecto de maiores dimensões. O segundo é o de determinar a forma como devem ser alocados os itens pequenos aos objectos grandes de forma a optimizar um determinado critério. O critério mais frequente consiste na minimização do desperdício global. Esses dois problemas dão origem ao problema que é conhecido na literatura sob a

designação de *problema de corte*, ou *Cutting Stock Problem* (CSP) na terminologia anglo-saxónica.

Na sequência da resolução do problema de corte, podem surgir outros problemas (auxiliares) ligados a diferentes aspectos de gestão. Um deles consiste em redefinir os padrões de corte de forma a garantir que o número de padrões diferentes seja o menor possível. Cada vez que é cortado um novo padrão, é necessário acertar as facas da máquina de corte aos novos tamanhos a cortar. Essa operação de inicialização da máquina (*setup*) leva tempo e pode ocasionalmente gerar desperdícios se for necessário efectuar testes. O problema de optimização associado é conhecido por problema de minimização de padrões, ou *Pattern Minimization Problem* (PMP) na terminologia anglo-saxónica. É esse problema específico que analisamos nesta dissertação.

Os problemas auxiliares ligados ao problema de corte são normalmente mais difíceis que o problema de corte standard. A título de exemplo, podemos citar o problema de minimização de pilhas abertas que consiste em determinar o plano de corte com o menor número de encomendas tratadas em simultâneo.

1.1 Os problemas de Corte e Empacotamento

Os problemas de corte e empacotamento estão fortemente ligados. A relação resulta da dualidade entre as noções de material e de espaço, *i.e.* entre o corpo do material sólido e o espaço ocupado por ele. Um corte pode ser visto como o empacotamento do espaço ocupado pelos itens mais pequenos no espaço ocupado pelos itens maiores. Por outro lado, o empacotamento pode ser visto como o corte do espaço ocupado pelos itens maiores em partes que correspondem aos itens pequenos.

A grande variedade de classificações e problemas tratados na literatura motivaram o desenvolvimento de tipologias por parte de investigadores ligados a esta área (D90, WHS07). Ambas as tipologias são extensas, e pretendem ser exaustivas. Sem entrarmos nos vários detalhes das tipologias, podemos dizer que os problemas de corte e empacotamento são classificados nesses trabalhos segundo as seguintes características essenciais:

- a dimensionalidade do problema: os problemas podem ir até três dimensões (o problema a uma dimensão considera figuras que podem ser reduzidas a segmentos de recta; o problema a duas dimensões trata do corte ou empacotamento de figuras planas e, finalmente, os problemas a três dimensões envolvem o tratamento de volumes);

- o tipo de afectação: os problemas podem tratar o caso em que todos os itens pequenos devem ser cortados ou empacotados de forma eficiente, *i.e.* minimizando o número de objectos grandes que são usados ou o seu valor; por outro lado, os problemas podem limitar o número de objectos grandes que podem ser usados obrigando a uma selecção dos itens pequenos que devem ser cortados ou empacotados (o critério mais frequente no qual se baseia essa selecção é a maximização do valor dos itens pequenos);

- o tipo de itens pequenos considerados: os problemas poderão ter um conjunto de itens pequenos fortemente ou fracamente heterogéneos; os problemas de empacotamento são tipicamente problemas fortemente heterogéneos com itens cuja procura é muito pequena, enquanto os problemas de corte correspondem aos casos em que os itens têm procuras altas;

- o tipo de objectos grandes considerados: esse objecto poderá ser único, ou poderão existir vários objectos diferentes;

- a forma geométrica dos itens pequenos: esses itens poderão ser figuras regulares como os rectângulos ou os círculos, ou figuras irregulares (peças de vestuário ou de calçado por exemplo).

A primeira formulação matemática do problema de corte foi apresentada por Kantorovich em 1939, apesar de só ter sido publicada em inglês em 1960 [K60]. Nos últimos 50 anos, os problemas de corte têm merecido a atenção de um vasto grupo de investigadores. Existe actualmente um grupo de interesse europeu dedicado a este assunto, e que é conhecido por European Interested Group on Cutting and Packing (ESICUP). Ao longo das últimas décadas, foram apresentados diferentes formulações e métodos de resolução exacta e heurística para os problemas de corte e empacotamento. Actualmente, muitos investigadores têm-se dedicado ao

estudo de variantes complexas destes problemas. Nesta dissertação, estudamos uma dessas variantes para a qual poucas abordagens exactas foram apresentadas até agora.

1.2 O Problema de Minimização de Padrões

O critério que é mais frequentemente usado em processos de corte é o da minimização dos desperdícios gerados. Não é no entanto o único critério. Em qualquer ambiente industrial, o corte propriamente dito é acompanhado por outras operações de gestão de stocks ou de configuração das máquinas. A nível dos stocks, por exemplo, um dos critérios frequentemente usados na indústria corresponde a minimizar o número de encomendas que são processadas ao mesmo tempo. Junto às máquinas de corte existem zonas de armazenamento, normalmente designadas por pilhas devido à forma como os objectos cortados são aí armazenados. Os itens relativos a uma encomenda são empilhados juntamente para facilitar a sua remoção, transporte e tratamento. Geralmente, o número de pilhas que podem ser constituídas é menor que o número de encomendas processadas. Se os padrões de corte não forem sequenciados correctamente, poderá não haver espaço para armazenar os itens que acabam de ser cortados. A única solução nesses casos passa por misturar os itens de encomendas distintas, o que não é de todo desejável.

Outro aspecto importante num processo de corte tem a ver com a configuração da própria máquina de corte. Quando um novo padrão de corte entra em produção, é necessário posicionar as facas segundo as dimensões dos itens a cortar. Quando o volume de encomendas é grande, o número de padrões diferentes pode ser também muito elevado. Nessas situações, o tempo perdido a acertar as facas da máquina pode constituir um verdadeiro problema. Por outro lado, poderão também existir custos associados a essa configuração. Em alguns casos, pode ser necessário efectuar testes para confirmar que as facas foram posicionadas correctamente. Esses testes podem gerar desperdícios que poderiam ser evitados, ou pelo menos reduzidos.

A Figura 1.1 representa uma instância do problema de corte standard a uma dimensão. Uma instância consiste num conjunto de m itens de comprimento w_i e procura b_i (i=1,...,m) que devem ser cortados a partir de rolos de comprimento único W. Todos os objectos têm a mesma altura, razão pela qual não é considerada essa dimensão.

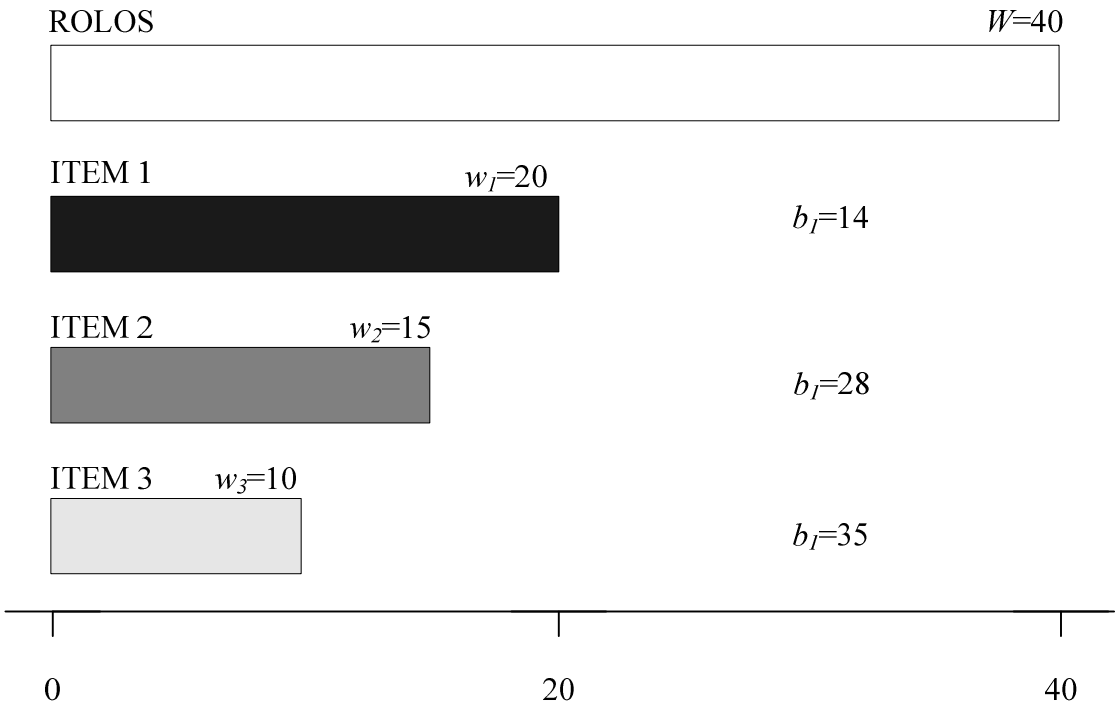

Figura 1.1: Instância do problema de corte standard a uma dimensão

A Figura 1.2 mostra como poderão ser cortados os itens da instância a partir dos rolos de comprimento W de modo a satisfazer todas as procuras. Cada configuração representa um padrão de corte. Por sua vez, o conjunto de todos os padrões de corte representa o plano global de corte. Na solução representada na Figura 1.2, os itens são cortados a partir de 28 rolos e de 4 padrões diferentes. Cada padrão j (j=1,...,m), é cortado exactamente 7 vezes (x_j). A máquina de corte terá de ser reajustada sempre que um novo padrão é cortado. Consequentemente, a máquina terá ser reconfigurada 4 vezes neste caso.

Se considerarmos uma troca simples de itens entre o primeiro e o segundo padrão, podemos reduzir o número de padrões diferentes no plano de corte. Os dois padrões são ambos cortados 7 vezes. Se substituirmos o item de comprimento 15 no primeiro padrão pelo item de comprimento 20 do segundo padrão, obtemos os dois padrões válidos seguintes:

PADRÃO A: $20 + 20 \leq W = 40$

PADRÃO B: $15 + 10 + 10 \leq W = 40$

O padrão B corresponde ao padrão 4 da Figura 1.2. O plano de corte que resulta dessa troca tem apenas 3 padrões diferentes, sendo que um deles é cortado um número mais elevado de vezes. No final, o número de rolos usados mantém-se inalterado. O respectivo plano de corte é ilustrado na Figura 1.3.

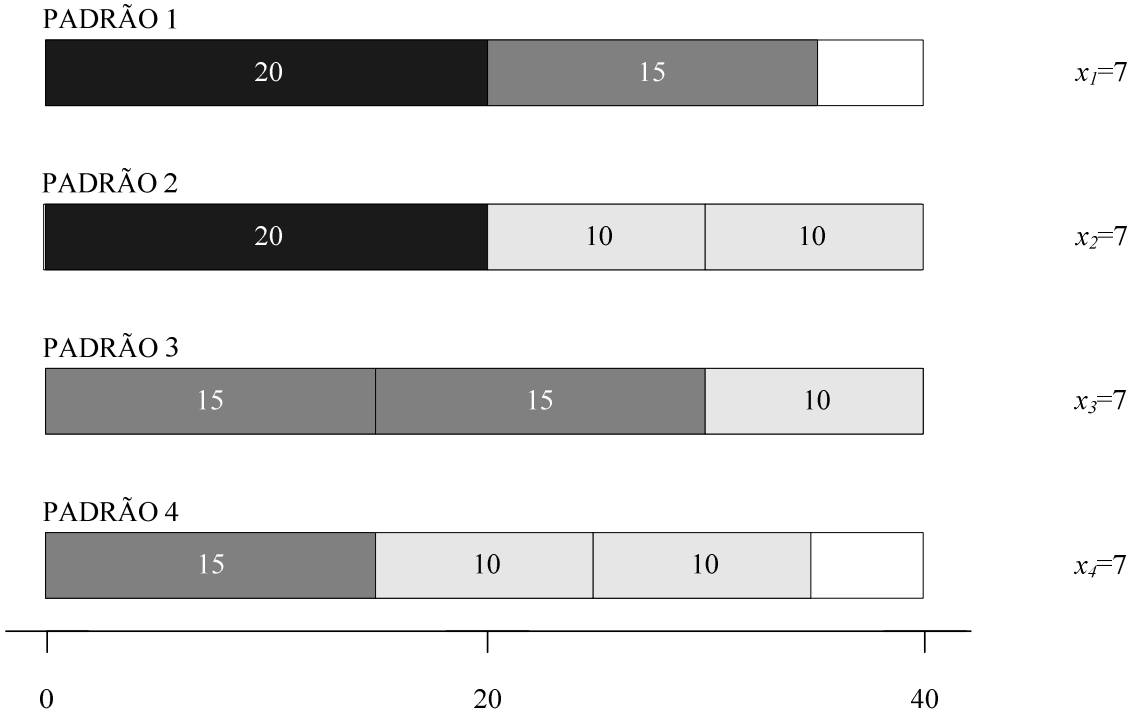

Figura 1.2: Plano de corte com 4 *setups*

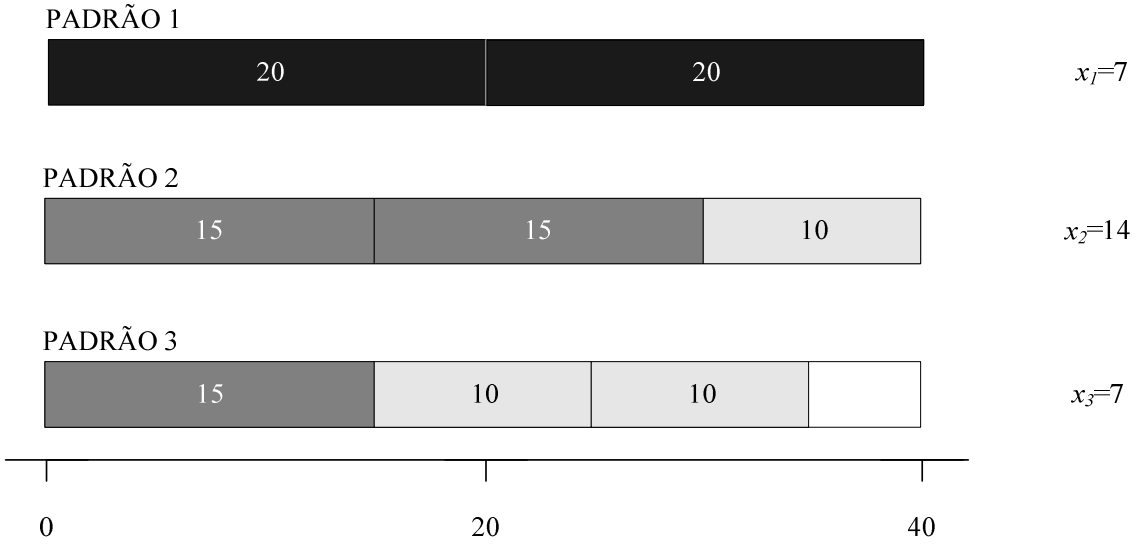

Figura 1.3: Plano de corte com 3 *setups*

Alguns métodos de resolução descritos na literatura procuram reduzir o número de padrões distintos baseando-se em trocas desse género entre itens de padrões diferentes.

O PMP pode ser abordado de duas formas distintas. A primeira consiste em resolver o problema considerando explicitamente o critério de minimização de desperdícios juntamente com o do número de padrões distintos. Nesse caso, o problema é resolvido numa única fase, e procura-se encontrar uma solução de equilíbrio entre o desperdício gerado e o número de operações de *setup* associadas a um plano de corte. A segunda abordagem consiste em resolver em primeiro lugar um problema de corte standard de modo a determinar o plano de corte que minimiza o desperdício total. De seguida, o problema de minimização de padrões é resolvido tendo como base o número de rolos obtido previamente. Nesse caso, o problema é resolvido em duas fases. Nesta dissertação consideramos este segundo caso.

Uma instância do PMP a uma dimensão que consideramos nesta dissertação é definida através dos elementos seguintes: um conjunto de m itens de comprimento w_i e procura b_i; um conjunto de rolos de comprimento único W, e cuja disponibilidade é assumida como sendo ilimitada; um número óptimo de rolos z_{CSP} obtido resolvendo o problema de corte standard associado. Ao longo desta dissertação, usaremos esta terminologia.

1.3 Abordagens de Resolução

Neste trabalho, consideramos a resolução exacta do Problema de Minimização de Padrões. Recorremos a métodos de Programação Inteira baseados em modelos de geração de colunas, no método de partição e avaliação sucessivas e no método de planos de corte. A maior parte das abordagens descritas na literatura baseia-se em métodos heurísticos de resolução. O número de algoritmos exactos é muito reduzido, e ainda só permitem resolver instâncias de pequena ou média dimensão.

Os métodos de geração de colunas baseiam-se no princípo da decomposição de modelos de Programação Linear de Dantzig e Wolfe. Dessa decomposição resultam modelos que são compostos por um número exponencial de variáveis ou colunas. Esses modelos são resolvidos normalmente gerando de forma dinâmica as colunas do modelo à medida que vão sendo

necessárias para melhorar o valor da função objectivo. Existem alguns modelos de geração de colunas para o PMP descritos na literatura. O primeiro modelo formulado explicitamente é da autoria de Vanderbeck [V00]. O autor baseia a sua decomposição num modelo original de afectação, em que as variáveis representam a associação de itens a rolos. O modelo original e a sua decomposição são não-lineares. Uma alternativa a esses modelos consiste numa extensão do modelo de geração de colunas para o problema standard de corte a 1-dimensão baseado na enumeração dos padrões de corte válidos [GG61]. Vários autores analisaram recentemente esse modelo. Compararam-o ao modelo de Vanderbeck e provaram que podia ser tão forte quanto esse quando são satisfeitas algumas condições.

Esta dissertação baseia-se num modelo de geração de colunas proposto recentemente por Alves *et al.* [AMVC09]. Esse modelo baseia-se numa nova decomposição do modelo original de afectação. A geração das colunas atractivas é feita a partir da resolução de dois subproblemas de mochila com restrições adicionais. Os autores analisaram a qualidade do modelo, e mostraram que podia ser mais forte que os anteriores. Nesta dissertação, analisamos estratégias para determinar soluções inteiras desse modelo, e discutimos também formas de refoçar o modelo.

Para determinar soluções inteiras, recorremos ao método de partição e avaliação sucessivas. Esse método é bem conhecido no domínio da Programação Inteira, e consiste em explorar uma árvore de pesquisa em que os nodos são subproblemas do problema original. No nosso caso, os problemas nos nodos são problemas de Programação Linear (relaxações lineares). Como o modelo que usamos é um modelo de geração de colunas, em cada nodo da árvore, temos de gerar colunas que possam ser atractivas, e que não estejam já no problema mestre restrito. Se não o fizermos, a solução inteira que será determinada poderá não ser óptima para o problema original. Com efeito, as colunas geradas no início do processo não tomam em conta as restrições adicionais de partição que são adicionadas aos subproblemas. A combinação do método de geração de colunas e do método de partição e avaliação sucessivas é designada por método de partição e geração de colunas (*branch-and-price*, na terminologia anglo-saxónica). Combinar os dois métodos não é fácil. A adição de restrições de partição aos modelos de geração de colunas pode destruir a estrutura dos subproblemas, tornando-os mais difíceis de resolver. Essa situação deve ser evitada tanto quanto possível. A resolução de um modelo de geração de colunas implica a resolução de um ou vários subproblemas de optimização em cada iteração. Quanto mais fácil for a resolução dos subproblemas, mais

eficiente será a resolução do modelo de geração de colunas. Para evitar que os subproblemas percam a sua estrutura inicial, as restrições de partição deverão ser definidas com base nas variáveis dos modelos originais.

Finalmente, a adição de planos de corte em modelos de Programação Inteira permite reforçar a qualidade dos modelos. A diferença entre um plano de corte e uma restrição de partição reside no facto do plano de corte não excluir nenhuma solução inteira válida, o que não acontece com as restrições de partição. Contudo, quando qualquer uma dessas restrições é adicionada a um modelo de geração de colunas, o efeito é o mesmo em termos do impacto na estrutura dos subproblemas. Assim, ao adicionar planos de corte a modelos de geração de colunas, deve-se ter cuidado em manter os subproblemas tão simples de resolver como o eram inicialmente. Nesta dissertação, exploramos alguns planos de corte que permitem aumentar a qualidade do limite inferior do modelo de Alves *et al.*[AMVC09].

1.4 Estrutura da Dissertação

A dissertação divide-se da seguinte forma. No Capítulo 2, analisamos os resultados descritos na literatura relativos ao Problema de Minimização de Padrões. Descrevemos em particular as abordagens exactas que foram propostas até agora.

O Capítulo 3 descreve em detalhe um modelo de Programação Inteira descrito recentemente na literatura, no qual se baseiam os resultados descritos nesta dissertação. Várias estratégias de reforço do modelo são discutidas.

No Capítulo 4, descrevemos procedimentos para a obtenção de soluções óptimas inteiras. A nossa abordagem baseia-se no método de partição e geração de colunas. Várias estratégias de partição são apresentadas. As estratégias foram desenvolvidas de forma a não aumentarem a dificuldade de resolução dos subproblemas de geração de colunas.

No Capítulo 5, apresentamos resultados computacionais para as abordagens descritas no Capítulo 4. Nesses testes, usámos instâncias reais do Problema de Minimização de Padrões que foram também usadas na literatura.

A dissertação termina com o Capítulo 6 onde se tiram várias conclusões relativamente ao trabalho realizado e aos resultados obtidos.

Capítulo 2

Modelos e Algoritmos para o Problema de Minimização de Padrões

2.1 Introdução

O critério que prevalece quando se determina um plano de corte é normalmente o da minimização de desperdícios. No entanto, esse não é o único factor operacional relevante. O modo como é executado o plano de corte, a definição estrutural dos padrões, a sequência em que eles são executados têm implicações a nível dos custos, sejam eles de *setup*, de matéria-prima, ou de gestão de inventários. Como vimos anteriormente, nesta dissertação analisaremos o caso particular em que se pretende minimizar o número de padrões diferentes no plano de corte. Assumimos que o número de objectos grandes é fixo, e igual à solução óptima do problema de corte correspondente. A solução do PMP é também aquela que tem associado o menor número de operações de inicialização ou *setups*.

Neste Capítulo, apresentamos os diferentes modelos e algoritmos descritos na literatura para o PMP. O Capítulo está dividido em duas partes. Na primeira, analisamos os métodos de resolução exacta baseados em modelos de Programação Inteira. Os métodos que exploramos consistem essencialmente em algoritmos de partição e avaliação sucessivas, e em métodos de planos de corte. Na segunda parte deste Capítulo, analisamos algumas das heurísticas mais eficientes que foram descritas na literatura. Dada a complexidade inerente ao problema, a grande maioria dos algoritmos para o PMP são do tipo heurístico. O número de algoritmos exactos que foram desenvolvidos até hoje é muito reduzido. Com os resultados desta

dissertação que descrevemos em próximos capítulos, esperamos contribuir para a resolução exacta do problema.

2.2 Modelos e Algoritmos Exactos

Existem apenas três métodos exactos descritos na literatura para a resolução do PMP. O primeiro foi proposto por Vanderbeck em [V00], e consiste num algoritmo de partição e geração de colunas com cortes. Vanderbeck definiu um modelo de geração de colunas aplicando o método da decomposição de Dantzig-Wolfe a um modelo compacto não linear. Alves e Carvalho [AVC08] usaram o mesmo modelo mas definiram um algoritmo de partição e geração de colunas robusto que provou ser mais eficiente que o método proposto por Vanderbeck. Adicionalmente, os autores propuseram também um novo modelo de fluxos em rede para o PMP. Belov [B03] usou um algoritmo baseado num modelo para o PMP que deriva do modelo de Gilmore e Gomory para o problema de corte. Esse modelo tem um número exponencial de linhas e colunas, e um limite contínuo de fraca qualidade. Belov usa cortes para reforçar esse modelo, mas mesmo assim não consegue superar nenhum dos dois algoritmos referidos acima. Nas próximas secções, revemos cada um desses métodos em detalhe.

2.2.1 Modelo Não-Linear de Vanderbeck

Em [V00], Vanderbeck apresenta um modelo original para o PMP que consiste num modelo quadrático compacto e não-linear com variáveis inteiras gerais e binárias. O modelo usa variáveis de afectação x_{ik} que associa os itens i aos padrões k, variáveis inteiras z_k que representam o nível de utilização dos padrões k , e variáveis binárias y_k que indicam se um padrão k é usado ou não. O modelo é formulado da seguinte forma:

$$\min z = \sum_{k=1}^{z_{CSP}} y_k \tag{1}$$

sujeito a

$$\sum_{k=1}^{z_{CSP}} z_k x_{ik} = b_i, \qquad i=1,\ldots,m, \tag{2}$$

$$\sum_{k=1}^{z_{CSP}} z_k \leq z_{CSP}, \tag{3}$$

$$z_k \leq z_{CSP} y_k, \qquad k=1,\ldots,z_{CSP}, \tag{4}$$

$$\sum_{i=1}^{m} w_i x_{ik} \leq W y_k, \qquad k=1,\ldots,z_{CSP}, \tag{5}$$

$$x_{ik} \in \mathbb{N}, \qquad i=1,\ldots,m,\ k=1,\ldots,z_{CSP}, \tag{6}$$

$$y_k \in \{0,1\}, \qquad k=1,\ldots,z_{CSP}, \tag{7}$$

$$z_k \in \mathbb{N}, \qquad k=1,\ldots,z_{CSP}. \tag{8}$$

As variáveis y_k estão associadas ao uso de um determinado padrão k. O objectivo consiste assim em minimizar a soma dessas variáveis para todos os padrões k. No máximo, existirão z_{CSP} padrões distintos, um para cada rolo que é usado. A função objectivo do modelo é expressa através da expressão (1). Apesar de poderem existir dois padrões iguais com índices k distintos, pode ser facilmente provado que a solução óptima não irá conter nenhum padrão nessas condições. Com efeito, se tivermos dois padrões iguais com índices k' e k'', e se $y_{k'}$ e $y_{k''}$ forem iguais a 1, fazendo $z_{k'}=z_{k'}+z_{k''}$ e $y_{k''}=0$, obtemos uma solução com um menor valor da função objectivo.

As restrições (2) asseguram que as procuras são satisfeitas exactamente. Essas restrições são não-lineares. O conjunto de restrições (3) limita o número de rolos que podem ser usados a z_{CSP}. As restrições (4) relacionam as variáveis y_k com as variáveis z_k. Assim, se um padrão k for usado ($z_k > 0$), teremos que $y_k = 1$. As restrições (5) são restrições de mochila que impedem os padrões de terem um comprimento superior ao comprimento W dos rolos. O modelo tem um número reduzido de restrições ($2\ z_{CSP} + m + 1$), e um número pseudo-polinomial de variáveis dado que dependem do parâmetro z_{CSP} do problema. Por essa razão, o modelo é designado por modelo compacto.

A não-linearidade do modelo constitui uma das suas principais fraquezas. Conforme apontado por Vanderbeck (V00), linearizar o modelo conduziria a um modelo cuja relaxação linear seria de fraca qualidade. Além disso, é sabido que as formulações baseadas em variáveis de afectação para os problemas de corte possuem um elevado grau de simetria o que

penaliza fortemente a sua resolução através de métodos como o de partição e avaliação sucessivas. O que acontece é que soluções que são iguais na prática podem ocorrer em nodos distintos sob uma representação diferente.

2.2.2 Modelo de Geração de Colunas de Vanderbeck

Dada a complexidade da resolução do modelo (1)-(8), e o facto de uma linearização conduzir a um modelo de fraca qualidade, Vanderbeck tenta resolver o PMP em [V00] através de uma reformulação baseada no princípio da decomposição de Dantzig-Wolfe. As decomposições desse género resultam na definição de um problema mestre composto por um número exponencial de colunas, e por um ou mais subproblemas através das quais podem ser geradas as colunas. Em [V00], Vanderbeck define o problema mestre com base nas restrições (2) e (3) do modelo da Secção anterior. As restantes restrições definem o subproblema. O problema mestre é definido da seguinte forma:

$$\min z_V = \sum_{k\in K}\sum_{n=1}^{u_k} \lambda_{kn} \tag{9}$$

sujeito a

$$\sum_{k\in K}\sum_{n=1}^{u_k} n a_{ik}\lambda_{kn} = b_i, \qquad i = 1,\dots,m, \tag{10}$$

$$\sum_{k\in K}\sum_{n=1}^{u_k} n\lambda_{kn} \le z_{CSP}, \tag{11}$$

$$\lambda_{kn} \in \{0,1\}, \qquad k \in K, n = 1,\dots,u_k. \tag{12}$$

As variáveis λ_{kn} do modelo estão associadas a padrões *k* do conjunto de todos os padrões designado por *K*. Essas variáveis são binárias. O número de vezes que um determinado padrão *k* é usado é representado pelo parâmetro *n*. As colunas do modelo correspondem assim a padrões válidos de corte com uma multiplicidade *n* associada. O parâmetro a_{ik} representa o número de vezes que um item *i* é incluído no padrão *k*.Como dissemos, as restrições (10) e (11) do modelo correspondem respectivamente às restrições (2) e (3) do modelo original.

A constante u_k representa a multiplicidade máxima que um padrão k pode tomar. Dada que as procuras devem ser satisfeitas exactamente, a multiplicidade de um padrão deverá ser tal que as procuras dos itens nunca seja excedida:

$$u_k = min_{i=1,\dots,m} \left\lfloor \frac{b_i}{a_{ik}} \right\rfloor.$$

Em [V00], Vanderbeck descreve também um limite máximo para o valor de todos os u_k. Essa multiplicidade máxima que vamos designar por n^{max} pode ser expressa da forma seguinte:

$$n^{max} = min\{z_{CSP} - LB_{PMP} + 1, max_i b_i\}.$$

O valor LB_{PMP} representa um limite inferior para o PMP que pode ser calculado por exemplo resolvendo o problema de empacotamento (*bin-packing*) correspondente, no qual todas as procuras dos itens foram fixadas a 1.

O subproblema é definido pelas restrições (4) e (5) do modelo não-linear, *ie* as restrições que limitam o número de vezes que um padrão pode ser usado (o número total de rolos) e as restrições de mochila. A formulação do subproblema é a seguinte:

$$\max z_{SP} = n\left(\sum_{i=1}^{m} \pi_i \, x_i - \delta\right) \tag{9}$$

sujeito a

$$\sum_{i=1}^{m} w_i \, x_i \leq W, \tag{10}$$

$$nx_i \leq b_i, \qquad i = 1, \dots, m, \tag{11}$$

$$n \in \mathbb{N}, \tag{12}$$

$$x_i \in \mathbb{N}, \qquad i = 1, \dots, m. \tag{13}$$

O modelo é composto pela variável n que representa o número de vezes que um padrão é usado (multiplicidade), e pelas variáveis x_i que representam o número de vezes que um item i é incluído no padrão. Os parâmetros π_i e δ são respectivamente os valores duais associados às restrições (10) e (11) do problema mestre.

As não-linearidades estão agora no subproblema. Conforme indicado em [V00], é possível resolver o subproblema enumerando cada um dos valores possíveis para n. Quando fixamos o valor de n, o problema de optimização resultante é um problema de mochila clássico que pode ser resolvido por exemplo usando Programação Dinâmica.

O Exemplo seguinte ilustra o modelo de geração de colunas de Vanderbeck.

Exemplo 2.1 Considere uma instância do PMP com rolos de comprimento W=15, e cujos itens são definidos da seguinte forma:

i	w_i	b_i
1	9	5
2	7	3
3	5	4
4	4	3
5	2	2

A solução óptima do problema de corte associado usa 8 rolos, o que corresponde a um desperdício total de 18 unidades:

$$D = 8 \times 15 - (9 \times 5 + 7 \times 3 + 5 \times 4 + 4 \times 3 + 2 \times 2)$$

Sendo uma instância de pequena dimensão, podemos enumerar o conjunto dos padrões válidos de corte. Esses padrões são ilustrados na Tabela 2.1. A coluna "utilização" da Tabela indica o número de vezes que o padrão é usado na solução óptima. O desperdício associado a cada padrão é indicado na coluna da direita. Os padrões correspondem às colunas do modelo de Gilmore e Gomory para o problema de corte standard a uma dimensão [GG61]. Os valores em cada uma das linhas indicam o número de vezes que um item é incluído no respectivo padrão de corte.

A solução representada na Tabela 2.1 usa exactamente 8 padrões diferentes. Esse valor constitui um limite superior para o valor óptimo do respectivo PMP. Um limite inferior para o PMP pode ser obtido através do problema de empacotamento com um rolo de comprimento W=15, e o mesmo conjunto de itens mas com procuras iguais a 1. Nesse caso, todos os

padrões são diferentes, e o número de rolos usados é igual ao número de padrões diferentes na solução. O valor da solução óptima do problema de empacotamento para a instância usada neste Exemplo é igual a 2.

Padrão *k*	**Utilização**	**Item *i***					**Desperdício**
		1	**2**	**3**	**4**	**5**	
		Quantidades a produzir					
1	0	1	0	0	0	0	6
2	**3**	1	0	1	0	0	1
3	**2**	1	0	0	1	0	2
4	0	1	0	0	1	1	0
5	0	1	0	0	0	1	4
6	0	1	0	0	0	2	2
7	0	0	1	0	0	0	8
8	0	0	2	0	0	0	1
9	**1**	0	1	1	0	0	3
10	0	0	1	1	0	1	1
11	**1**	0	1	0	1	0	4
12	0	0	1	0	1	1	2
13	0	0	1	0	1	2	0
14	0	0	1	0	2	0	0
15	0	0	1	0	0	1	6
16	**1**	0	1	0	0	2	4
17	0	0	0	1	0	0	10
18	0	0	0	2	0	0	5
19	0	0	0	3	0	0	0
20	0	0	0	1	1	0	6
21	0	0	0	1	2	0	2
22	0	0	0	1	0	1	8
23	0	0	0	1	0	2	6
24	0	0	0	2	1	0	1
25	0	0	0	2	0	1	3
26	0	0	0	2	0	2	1
27	0	0	0	1	1	1	4
28	0	0	0	1	1	2	2
29	0	0	0	1	2	1	0
30	0	0	0	0	1	0	11
31	0	0	0	0	2	0	7
32	0	0	0	0	3	0	3
33	0	0	0	0	1	1	9
34	0	0	0	0	1	2	7
35	0	0	0	0	2	1	5
36	0	0	0	0	2	2	3
37	0	0	0	0	3	1	1
38	0	0	0	0	0	1	13
39	0	0	0	0	0	2	11

Tabela 2.1: Conjunto de padrões de corte válidos (Exemplo 2.1)

O intervalo no qual se encontra o óptimo do PMP é o seguinte:

$$z_V^* \in [2; 8]$$

O problema mestre do modelo de geração de colunas de Vanderbeck depende do conjunto de padrões válidos. Para cada padrão de corte, existe um conjunto de colunas no modelo de Vanderbeck. Cada subconjunto de colunas corresponde a um determinado padrão de corte com uma multiplicidade associada. A Tabela 2.2 ilustra a estrutura dessas colunas. Nessa Tabela, listam-se todos os pares padrões e multiplicidade válidos. A coluna *n* representa a multiplicidade do padrão. Quando *n=1*, o padrão corresponde exactamente ao padrão de corte de mesmo índice indicado na Tabela 2.1. A coluna (*k*,*n*) representa o par padrão/multiplicidade. Quanto ao desperdício associado a um padrão, ele corresponde agora ao desperdício do padrão original de corte multiplicado pela multiplicidade associada.

A solução óptima inteira é também representada nessa Tabela. O problema pode ser resolvido usando no mínimo 3 padrões distintos, ou seja menos de metade dos padrões usados na solução do problema de corte descrita na Tabela 2.1. A solução óptima do PMP é representada na Figura 2.1. Os valores 1 no final dos padrões representam unidades de deseperdício. No final, a solução tem obviamente o mesmo desperdício total do que a solução dada pelo problema de corte.

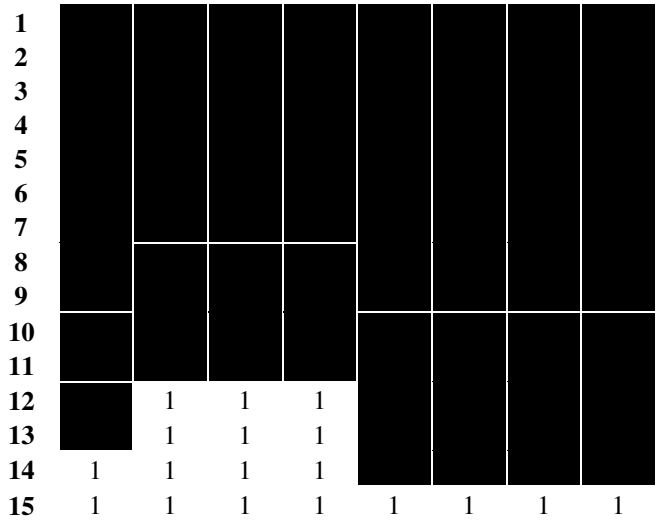

Figura 2.1: Solução óptima do PMP (Exemplo 2.1)

Padrão k	(k,n)	Utilização	Item i					n	Desperdício
			1	2	3	4	5		
			Quantidades a produzir						
1	1.1	0	1	0	0	0	0	1	6
	1.2	0	2	0	0	0	0	2	12
	1.3	0	3	0	0	0	0	3	18
	1.4	0	4	0	0	0	0	4	24
	1.5	0	5	0	0	0	0	5	30
2	2.1	0	1	0	1	0	0	1	1
	2.2	0	2	0	2	0	0	2	2
	2.3	0	3	0	3	0	0	3	3
	2.4	**1**	4	0	4	0	0	4	4
3	3.1	0	1	0	0	1	0	1	2
	3.2	0	2	0	0	2	0	2	4
	3.3	0	3	0	0	3	0	3	6
4	4.1	0	1	0	0	1	1	1	0
	4.2	0	2	0	0	2	2	2	0
5	5.1	0	1	0	0	0	1	1	4
	5.2	0	2	0	0	0	2	2	8
6	6.1	**1**	1	0	0	0	2	1	2
7	7.1	0	0	1	0	0	0	1	8
	7.2	0	0	2	0	0	0	2	16
	7.3	0	0	3	0	0	0	3	24
8	8.1	0	0	2	0	0	0	1	1
9	9.1	0	0	1	1	0	0	1	3
	9.2	0	0	2	2	0	0	2	6
	9.3	0	0	3	3	0	0	3	9
10	10.1	0	0	1	1	0	1	1	1
	10.2	0	0	2	2	0	2	2	2
11	11.1	0	0	1	0	1	0	1	4
	11.2	0	0	2	0	2	0	2	8
	11.3	**1**	0	3	0	3	0	3	12
12	12.1	0	0	1	0	1	1	1	2
	12.2	0	0	2	0	2	2	2	4
13	13.1	0	0	1	0	1	2	1	0
14	14.1	0	0	1	0	2	0	1	0
15	15.1	0	0	1	0	0	1	1	6
	15.2	0	0	2	0	0	2	2	12
16	16.1	0	0	1	0	0	2	1	4
17	17.1	0	0	0	1	0	0	1	10
	17.2	0	0	0	2	0	0	2	20
	17.3	0	0	0	3	0	0	3	30
	17.4	0	0	0	4	0	0	4	40
18	18.1	0	0	0	2	0	0	1	5
	18.2	0	0	0	4	0	0	2	10
19	19.1	0	0	0	3	0	0	1	0
20	20.1	0	0	0	1	1	0	1	6
	20.2	0	0	0	2	2	0	2	12
	20.3	0	0	0	3	3	0	3	18
21	21.1	0	0	0	1	2	0	1	2
22	22.1	0	0	0	1	0	1	1	8
	22.2	0	0	0	2	0	2	2	16
23	23.1	0	0	0	1	0	2	1	6
24	24.1	0	0	0	2	1	0	1	1
	24.2	0	0	0	4	2	0	2	2
25	25.1	0	0	0	2	0	1	1	3
	25.2	0	0	0	4	0	2	2	6
26	26.1	0	0	0	2	0	2	1	1
27	27.1	0	0	0	1	1	1	1	4
	27.2	0	0	0	2	2	2	2	8
28	28.1	0	0	0	1	1	2	1	2
29	29.1	0	0	0	1	2	1	1	0
30	30.1	0	0	0	0	1	0	1	11
	30.2	0	0	0	0	2	0	2	22
	30.3	0	0	0	0	3	0	3	33
31	31.1	0	0	0	0	2	0	1	7
32	32.1	0	0	0	0	3	0	1	3
33	33.1	0	0	0	0	1	1	1	9
	33.2	0	0	0	0	2	2	2	18
34	34.1	0	0	0	0	1	2	1	7
35	35.1	0	0	0	0	2	1	1	5
36	36.1	0	0	0	0	2	2	1	3
37	37.1	0	0	0	0	3	1	1	1
38	38.1	0	0	0	0	0	1	1	13
	38.2	0	0	0	0	0	2	2	26
39	39.1	0	0	0	0	0	2	1	11

Tabela 2.2: Conjunto de padrões de corte válidos e multiplicidades (Exemplo 2.1)

Como vimos, a formulação é linear mas com um número muito elevado de variáveis. Para a instância de muito pequena dimensão que consideramos neste Exemplo, o número de variáveis já é considerável. Esse valor cresce de forma exponencial à medida que aumenta o número de itens na instância. É por essa razão que Vanderbeck resolve o modelo usando geração diferida de colunas. Começa por resolver um problema mestre restrito (com um subconjunto de colunas), e gera as colunas iterativamente à medida que vão sendo necessárias.

O subproblema consiste em determinar um padrão de corte válido, *ie* cujo comprimento total seja inferior a 15, para cada valor possível de multiplicidade. Para o caso mais simples em que a multiplicidade é igual a 1, o subproblema consiste no seguinte problema de optimização.

$$\max z_{SP} = \sum_{i=1}^{m} \pi_i \, x_i - \delta$$

sujeito a

$$\sum_{i=1}^{m} w_i \, x_i \leq 15,$$

$$x_i \leq b_i, \qquad i = 1, \dots, m,$$

$$x_i \in \mathbb{N}, \qquad i = 1, \dots, m.$$

O segundo conjunto de restrições garante que os padrões são padrões próprios, *ie* que não têm mais itens do que aqueles que são procurados. Para n=2, o problema de optimização é essencialmente o mesmo (saco de mochila). Os limites superiores relativamente ao número de vezes que um item pode ser incluído num padrão é agora mais pequeno.

$$\max z_{SP} = \sum_{i=1}^{m} \pi_i \, x_i - \delta$$

sujeito a

$$\sum_{i=1}^{m} w_i \, x_i \leq 15,$$

$$x_i \leq \left\lfloor \frac{b_i}{2} \right\rfloor, \qquad i = 1, \dots, m,$$

$x_i \in \mathbb{N}, \qquad i = 1, \dots, m.$

À medida que a multiplicidade aumenta o subproblema vai ficando cada vez mais fácil. Alguns itens podem mesmo ser removidos da instância, uma vez que o seu limite superior é igual a 0. ■

Uma vez que a não-linearidade do modelo compacto passou para o subproblema de geração de colunas, é natural que o subproblema seja agora o ponto fraco da reformulação. Linearizando esse subproblema faz com que em cada iteração se tenha de resolver no máximo n^{max} problemas de mochila. Em [V00], Vanderbeck faz notar que na prática não é necessário resolver um problema de mochila para cada valor de multiplicidade de 1 até n^{max} dado que a solução óptima do problema de mochila se pode manter óptima para vários valores sucessivos de multiplicidade. Uma solução do problema de mochila mantém-se óptima até à multiplicidade:

$$n^* = min_i \left\{ \left\lfloor \frac{b_i}{x_i^*} \right\rfloor \right\},$$

em x_i^* representa o valor da variável x_i na solução óptima do problema de mochila..

Em [V00], Vanderbeck determina a solução óptima inteira do PMP usando um algoritmo de partição e geração de colunas com cortes. Usa o modelo (9)-(12) para determinar limites inferiores nos nodos da árvore de partição. O algoritmo é descrito em detalhe na próxima Secção.

2.2.3 Algoritmo de Partição e Geração de Colunas com Cortes de Vanderbeck

Conforme indica Vanderbeck [V00], a relaxação linear do modelo (9)-(12) continua a ser de fraca qualidade, não sendo tão forte como aquela que é possível obter para o problema de corte standard a uma dimensão. O intervalo médio de integralidade para as instâncias que Vanderbeck testou é em média de 33,5%. Por esse motivo, o autor recorre a planos de corte para reforçar a formulação.

Existe no entanto um problema que é necessário ter em consideração quando se tenta combinar planos de corte com modelos de geração de colunas, pois a introdução destes cortes pode implicar modificações na estrutura dos custos reduzidos das colunas. Vanderbeck evita esse problema gerando famílias de cortes a partir dos coeficientes de apenas uma restrição do modelo (9)-(12). Os cortes são derivados usando funções superaditivase não-decrescentes. Em [NW99], Nemhauser e Wolsey mostram que essas funções podem ser usadas para derivar inequações válidas para problemas de Programação Linear.

Uma função *f* é superaditiva se verificar a seguinte condição:

$$f(x) + f(y) \leq f(x + y).$$

Vanderbeck usa a seguinte função superaditiva e não-decrescente para derivar os planos de corte:

$$F^{\gamma}(z) = \max\left\{0, \left\lceil \frac{\gamma z}{b} \right\rceil - 1\right\},$$

em que *b* corresponde ao termo independente de uma restrição linear do tipo:

$$\sum_i a_i x_i \quad \leq b.$$

O plano de corte é definido a partir desta última restrição, à qual se aplica a função superaditiva indicada acima:

$$\sum_i F^{\gamma}(a_i) x_i \quad \leq F^{\gamma}(b).$$

Qualquer restrição linear pode ser usada para derivar esses planos de corte desde que os coeficientes a_i sejam positivos e menores do que o termo independente *b*.

Vanderbeck deriva os planos de corte a partir das restrições (10) e (11) do modelo de geração de colunas. Os cortes permitem melhorar significativamente a qualidade do modelo. O valor do limite inferior dado pela relaxação linear aumenta em média de 17,3%, e o intervalo de integralidade é reduzido para 13,8% em média.

O facto de derivar os cortes com base numa única restrição do problema mestre permite manter quase inalterada a estrutura do subproblema que continua a ser um problema de mochila. O tempo médio gasto em gerar colunas é de 15,9% do tempo total de computação para as instâncias reais testadas por Vanderbeck.

Os cortes não são suficientes para obter soluções óptimas inteiras. Para determinar uma solução inteira, Vanderbeck recorre ao método de partição e avaliação sucessivas. O autor sugere que a partição seja feita com base em propriedades que possam ser facilmente identificadas nos subproblemas. Vanderbeck define uma regra de partição baseada em 7 pontos. A ideia principal é a de estabelecer partições entre o conjunto de colunas com base no valor da multiplicidade das colunas e nos itens que compõem os padrões. Com essas regras de partição, Vanderbeck consegue resolver 12 das 16 instâncias reais que usa nas experiências computacionais descritas em [V00].

2.2.4 Modelo de Geração de Colunas baseado no Modelo de Gilmore e Gomory

O modelo de Gilmore e Gomory [GG61] para o problema de corte standard a uma dimensão pode ser usado para formular o PMP. A cada coluna do problema mestre associa-se uma variável binária que indica se o padrão é usado our não. O resultado é um modelo com um número exponencial de colunas e restrições:

$$\min z_{GG'} = \sum_{k \in K} \lambda_k \tag{13}$$

sujeito a

$$\sum_{k \in K} a_{ik} x_k = b_i, \qquad i = 1, \dots, m, \tag{14}$$

$$\sum_{k \in K} x_k \leq z_{CSP}, \tag{15}$$

$$x_k \leq z_{CSP} \lambda_k, \qquad k \in K, \tag{16}$$

$$x_k \in \mathbb{N}, \qquad k \in K, \tag{17}$$

$$\lambda_k \in \{0,1\}, \qquad k \in K. \tag{18}$$

As variáveis x_k representam o número de vezes que um padrão *k* é usado. Se um padrão *k* for usado, a variável binária λ_k será igual a 1. A associação entre as duas variáveis é representada através das restrições (16) do modelo. A função objectivo (13) consiste em minimizar o número de padrões diferentes. As restrições (14) obrigam a que a procura dos itens seja satisfeita exactamente. A restrição (15) não permite que sejam usados mais do que z_{CSP} rolos.

O número exponencial de colunas e restrições é um dos grandes inconvenientes desse modelo. Em [V00], Vanderbeck mostra que o limite inferior da relaxação linear é muito fraco, sendo igual a 1 qualquer que seja a instância considerada. Alguns anos mais tarde, Belov em [B03] mostrou contudo que o modelo pode ser facilmente reforçado substituindo a restrição (16) por

$$x_k \leq u_k \lambda_k, k \in K.$$

Nesse caso, o modelo é tão forte como o modelo de geração de colunas proposto por Vanderbeck. Em [B03], Belov usa um algoritmo de partição e geração de colunas para resolver o modelo (13)-(18). O autor consegue resolver apenas 8 das 16 instâncias usadas por Vanderbeck em [V00]. Por essa razão, não entraremos aqui nos detalhes do algoritmo proposto por esse autor.

2.2.4 Modelo de Fluxos de Alves e Carvalho

Em [AVC08], Alves e Carvalho definem um modelo de fluxos para o PMP. Os autores não resolvem directamente esse modelo, mas usam-no para derivar restrições de partição para o

modelo de geração de colunas de Vanderbeck. O algoritmo proposto por esses autores é descrito em detalhe na próxima Secção.

À semelhança do problema de corte standard a uma dimensão, o PMP pode ser formulado usando um modelo de fluxos. Para isso, usamos um grafo dirigido e acíclico designado por $G=(V,A)$. Os arcos do grafo representam itens do PMP colocados numa determinada posição do rolo, e com uma determinada multiplicidade associada. O desperdício é modelado através de arcos de comprimento igual a 1 que se concentram no final dos padrões. O conjunto A de arcos é dividido em subconjuntos A^n, um por cada multiplicidade n. As variáveis x_{ij}^n do modelo identificam o número de itens cujo comprimento é igual a j-i colocados na posição i do rolo num padrão cuja multiplicidade é igual a n. Um caminho no grafo representa um padrão de corte válido com uma determinada multiplicidade.

O modelo de fluxos para o PMP é o seguinte:

$$\min z_F = \sum_{n=1}^{n^{max}} z^n \tag{19}$$

sujeito a

$$-\sum_{(r,s)\in A^n} x_{rs}^n + \sum_{(s,t)\in A^n} x_{st}^n = \begin{cases} z^n, & \text{se } s=0, \\ -z^n, & \text{se } s=W, \\ 0, & \text{caso contrário,} \end{cases} \quad n=1,\dots,n^{max}, \tag{20}$$

$$\sum_{n=1}^{n^{max}} \sum_{(r,r+w_i)\in A^n} n x_{r,r+w_i}^n = b_i, \quad i=1,\dots,m, \tag{21}$$

$$x_{rs}^n \in \mathbb{N}, \quad n=1,\dots,n^{max}, (r,s)\in A^n \tag{22}$$

$$z^n \in \mathbb{N}, \quad n=1,\dots,n^{max}. \tag{23}$$

As variáveis z^n do modelo representam o número de padrões diferentes de multiplicidade n que são usados. As restrições (20) são restrições de conservação de fluxo entre arcos da mesma multiplicidade. As restrições (21) são as restrições de procura. O fluxo num arco do conjunto A^n contribui com um factor n para a procura do item ao qual está associado. Nesse modelo, o PMP consiste em determinar o fluxo com o menor custo entre 0 e W que garante que as procuras são todas satisfeitas.

Na Figura 2.2, representamos graficamente o modelo de fluxos através do respectivo grafo G para a instância do Exemplo 2.1. As diferentes cores na Figura estão associadas à multiplicidade do arco.

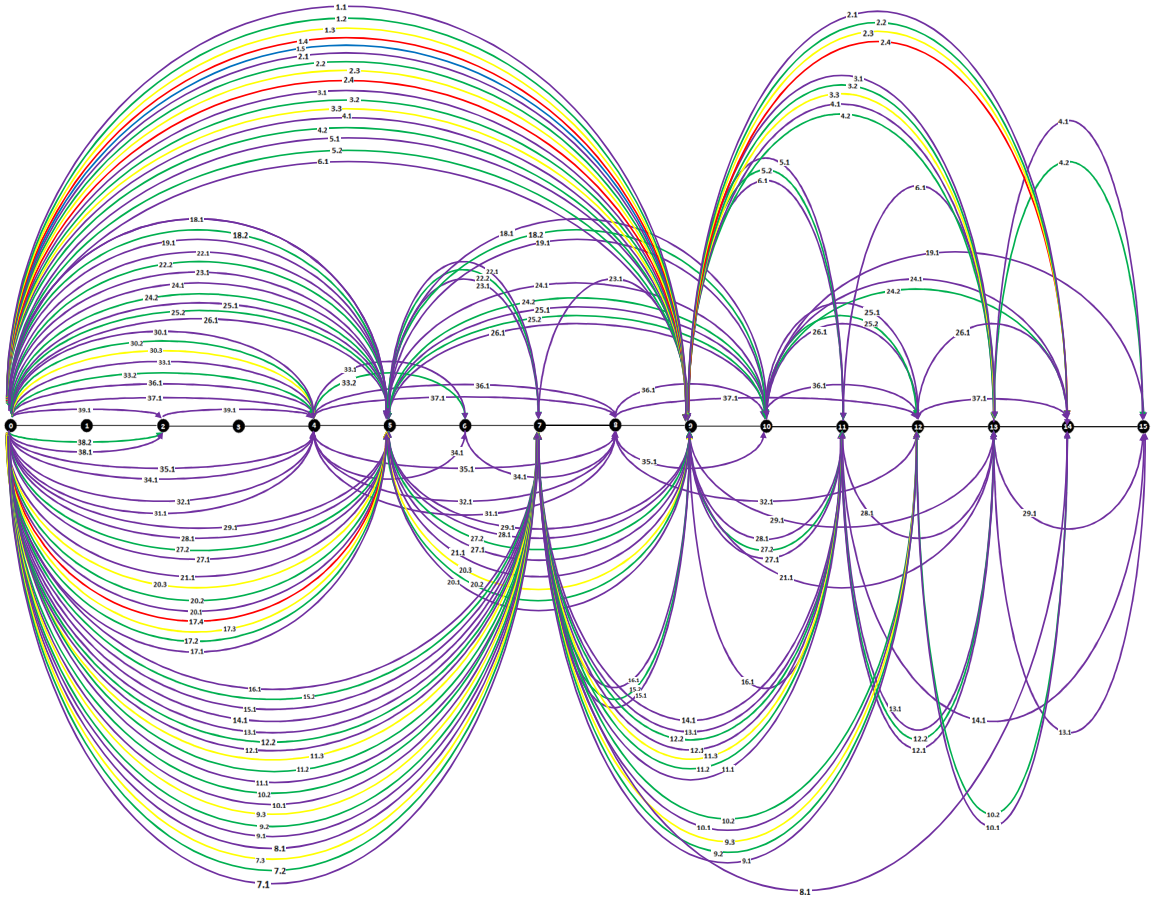

Figura 2.2: Grafo associado ao modelo de fluxos (Exemplo 2.1)

Como acontece com o modelo de fluxos para o problema de corte a uma dimensão, o modelo (19)-(23) tem um número pseudo-polinomial de restrições devido às restrições de conservação de fluxo que dependem de W. O modelo é útil porque permite derivar boas regras de partição [AVC08].

2.2.5 Algoritmo de Partição e Geração de Colunas com Cortes de Alves e Carvalho

Em [AVC08], Alves e Carvalho descrevem várias melhorias à abordagem que Vanderbeck descreve em [V00]. Os autores começam por introduzir uma nova restrição que limita o número de colunas do modelo de geração de colunas de Vanderbeck. Essa restrição depende do valor total do desperdício associado à solução óptima do problema de corte:

$$l = z_{CSP}W - \sum_{i=1}^{n} w_i b_i.$$

Todos as colunas do modelo de Vanderbeck que têm um desperdício superior a esse valor podem ser excluídas do problema mestre dado que nunca estarão na solução óptima inteira do problema. Assim, uma coluna é válida para o modelo de geração de colunas de Vanderbeck se e só se verificar o seguinte:

$$W - \sum_{i=1}^{n} w_i b_i \leq \left\lfloor \frac{l}{n} \right\rfloor.$$

Os autores mostram que essa restrição aumenta muito pouco o limite inferior do modelo, mas permite obter a solução através de um menor número de iterações do método de geração de colunas. A redução é em média de cerca de 34%.

Alves e Carvalho usam também pela primeira vez o conceito de funções duais válidas para derivar planos de corte para o modelo de Vanderbeck. Os autores mostram que algumas funções propostas na literatura são mais fortes que a função usada por Vanderbeck. Essas funções duais válidas tinham sido usadas anteriormente para derivar limites inferiores para o problema de corte e empacotamento [FS01]. Os autores mostraram que algumas dessas funções eram super-aditivas o que permite que elas sejam usadas para derivar planos de corte válidos par problemas de Programação Linear. Em [AVC08], Alves e Carvalho comparam os limites inferiores obtidos com as funções duais válidas com os resultados obtidos com os planos de corte de Vanderbeck. Os resultados em termos da qualidade do limite inferior melhoram com as funções usadas por esses autores.

Alves e Carvalho resolvem o PMP usando o método de partição e avaliação sucessivas combinado com o método de geração de colunas e de planos de corte. Os autores usam uma regra de partição baseada no modelo de fluxos apresentado atrás. Resolvem a relaxação linear do modelo de geração de colunas de Vanderbeck, e de seguida convertem essa solução numa sequência de fluxos. Se alguma variável de fluxo for fraccionária, particionam o problema criando dois subproblemas a partir das restrições seguintes:

$$x_{rs}^{n} \leq \lfloor x_{rs}^{n} \rfloor$$

e

$$x_{rs}^{n} \geq \lceil x_{rs}^{n} \rceil.$$

As restrições de partição são derivadas a partir de variáveis de um modelo original (que não foi decomposto), e como tal a regra é robusta porque não destroi a estrutura dos subproblemas. Os subproblemas em qualquer nodo da árvore de partição continuam a ser problemas de mochila com restrições adicionais.

Os autores testaram o algoritmo nas instâncias reais usadas em [V00] e em instâncias geradas aleatoriamente. Foram capazes de resolver mais uma instância (em 16) de Vanderbeck, mas sobretudo fazem-no usando menos nodos da árvore. Os autores lembram que não usam nenhuma heurística de arredondamento ao contrário do que faz Vanderbeck em [V00]. Os resultados apresentados em [AVC08] permitem a correctamente a eficiência da regra de partição e método de planos de corte usados.

2.3 Abordagens Heurísticas

Nesta Secção, analisamos algumas heurísticas descritas na literatura para resolver o PMP. Abordamos em particular as heurísticas que foram sugeridas por Haessler em [H75], por Foerster e Waescher em [FW00], e por Yanasse e Limeira em [YL06] por serem as heurísticas que estão ao nível do estado-da-arte.

Desenvolver uma boa heurística passa por identificar de forma acertada os elementos que caracterizam uma boa solução para um problema. No caso do PMP, devem ser privilegiados aqueles padrões cujo desperdício é reduzido, e cujo nível de utilização é o mais elevado possível. Os padrões com baixo desperdício podem mais facilmente serem repetidos várias vezes sem exceder o desperdício limite fixado pela solução óptima do problema de corte. Por outro lado, quanto maior for a utilização de um padrão menor será a necessidade de recorrer a outros padrões (diferentes) para satisfazer a procura dos itens restantes. As heurísticas que descrevemos a seguir tiram partido desta observação simples.

2.3.1 Heurística de Haessler

A heurística proposta por Haessler em [H75] permite resolver o problema de corte com um custo fixo associado à inicialização de um padrão. Trata-se de uma heurística sequencial em que, em cada iteração, se procura determinar padrões com baixos níveis de desperdícios, e altos níveis de utilização.

O procedimento de resolução começa com a análise das procuras dos vários itens que devem ser cortados. O objectivo é o de avaliar dois dos indicadores que servirão a caracterizar o problema. O primeiro indicador corresponde à estimativa do número de rolos necessários para cortar os itens pedidos, isto é:

$$\left\lceil \sum_i \frac{R_i' w_i}{W} \right\rceil,$$

em que W corresponde à largura dos rolos, w_i corresponde ao tamanho do item i, e R_i' representa a procura residual do item i, isto é o número de itens i que ainda faltam cortar. Essa estimativa é igual ao limite inferior trivial para o problema de corte a uma dimensão.

O segundo indicador permite avaliar o número médio de itens que deverá ser possível cortar a partir de cada rolo. Esse valor é calculado tendo como base a média do número de rolos calculada anteriormente, e é dado pela expressão seguinte:

$$\sum_i R_i' \Big/ \left\lceil \sum_i \frac{R_i' w_i}{W} \right\rceil.$$

Haessler usa esses indicadores para definir o que designa por *nível de aspiração* de um padrão. É com base nesse nível de aspiração que é possível determinar se um padrão deve ou não fazer parte da solução. Haessler faz depender o nível de aspiração dos seguintes parâmetros:

i) O Desperdício Máximo Permitido, DMP;
ii) O número mínimo e máximo de itens que podem ser cortados a partir de um simples rolo (MINP e MAXP, respectivamente);
iii) O número mínimo de vezes que um padrão pode ser utilizado, MINI.

Em [H75], o autor sugere valores de referência para esses parâmetros. Para o DMP, recomenda o intervalo seguinte de valores:

$$\lfloor 0.006W \rfloor \leq DMP \leq \lfloor 0.03W \rfloor.$$

Ao escolher este intervalo, Haessler favorece soluções com níveis de desperdício baixos, abaixo dos 3% por rolo. Obviamente, e como lembra o autor no seu artigo, esse requisito é relaxado à medida que a instância residual vai ficando mais pequena.

O valor de MAXP é escolhido tendo em conta a capacidade da máquina de corte no que diz respeito ao número máximo de itens que pode cortar de uma só vez. Para o valor de MINP, Haessler recomenda que seja usado o número médio de itens que devem ser cortados a partir de um simples rolo, ao qual subtraímos uma unidade*:*

$$\left\lceil \sum_i R_i' \Big/ \left\lceil \sum_i \frac{R_i' w_i}{W} \right\rceil \right\rceil - 1.$$

Para o valor de MINI, o autor sugere o seguinte intervalo

$$\left\lfloor 0.5 \sum_i R_i' \right\rfloor \leq MINI \leq \left\lfloor 0.9 \sum_i R_i' \right\rfloor.$$

A heurística procura assim determinar um padrão que cumpra o nível de aspiração. Em [H75], Haessler define formalmente esses requisitos da forma seguinte:

1. $W - \sum_i A_{ij} w_i \leq DMP$: o termo do lado esquerdo da inequação corresponde ao nível de desperdício de um padrão *j*, e esse não poderá ser superior ao DMP fixado anteriormente;
2. $MINP \leq \sum_i A_{ij} \leq MAXP$: A_{ij} representa o número de vezes que um item *i* é colocado no padrão *j*, e logo $\sum_i A_{ij}$ representa o número total de itens que estão no padrão *j*; esse valor terá de estar incluído no intervalo definido pelos valores MINP e MAXP;
3. $\frac{R_i'}{(A_{ij} \times MINI)} \geq 1 \ (A_{ij} > 0)$: o objectivo aqui é impedir que um item apareça demasiadas vezes num padrão limitando assim o número de vezes que ele possa ser usado, e prejudicando eventualmente a solução final.

Os padrões de corte são gerados por ordem lexicográfica decrescente. Apenas os padrões que incluam o item cuja procura residual é a mais elevada são considerados. Se a procura falhar tendo em conta os requisitos fixados (nível de aspiração), esses requisitos são relaxados. Esse processo é feito de forma iterativa até que um padrão válido seja identificado. Logo que um padrão que cumpra os requisitos seja identificado, ele é adicionado à solução, com um nível de utilização máximo. Os dados da instância são actualizados de modo a reflectir nas procuras a introdução na solução deste novo padrão. As procuras dos itens passam assim a ser procuras residuais. A heurística prossegue com essa instância residual, até não existirem mais itens por cortar.

2.3.2 Heurística de Foerster e Waescher

Em [FW00], Foerster e Waescher propõem uma nova heurística para o Problema de Minimização de Padrões, que consiste numa generalização de um método proposto previamente por Diegel *et al.* [D93] para este tipo de problemas.

Os métodos descritos em [FW00] são designados por métodos KOMBI. Esses métodos consistem em combinar padrões, de modo a reduzir o número de padrões diferentes, de tal modo que a soma dos níveis de utilização finais seja igual à soma dos níveis de utilização dos padrões originais. Só dessa forma se poderão manter os níveis de itens cortados constante.

Os métodos KOMBI têm a sua raiz no método de combinação 2 para 1 proposto por Diegel *et al.*([D93]). O objectivo desse método é de identificar dois padrões diferentes que possam ser agregados num só, e de determinar o nível de utilização do padrão resultante. O método é descrito na literatura da forma seguinte. É possível substituir dois padrões P1 e P2 por um padrão único P3, se o rácio entre o número de vezes que cada um dos itens é considerado em ambos os padrões e o nível de utilização agregado dos padrões for um valor inteiro para todos os itens considerados em P1 e P2. Formalmente, o número de vezes que um item *i* é considerado em P1 e P2 é dado pela expressão seguinte:

$$s_i = a_{i1}x_1 + a_{i2}x_2,$$

assumindo que os níveis de utilização de P1 e P2 são respectivamente iguais a x_1 e x_2, e que a_{i1} e a_{i2} representam também respectivamente o número de vezes que o item *i* é cortado no padrão P1 e P2. Assim, se

$$\frac{s_i}{x_1 + x_2}$$

for um inteiro para qualquer valor de *i*, então o padrão P1 e P2 poderão ser substituídos por um padrão P3, em que

$$a_{i3} = \frac{s_i}{x_1 + x_2}, \forall i, \text{ e } \quad x_3 = x_1 + x_2.$$

Foerster e Waescher começam por extender essas combinações a casos em que mais padrões são envolvidos. Consideram os casos das combinações 3 para 2, 3 para 1, 4 para 3, 2 e 1, e finalmente, o caso geral das combinações *p* para *q*. Em todos esses casos, considera-se a propriedade segundo a qual o nível de utilização dos padrões resultantes deve ser igual à soma dos níveis de utilização de 2 ou mais padrões originais.

Os métodos KOMBI combinam as diversas formas de agregar padrões diferentes. Os autores observam que a complexidade dessas combinações aumenta com o número padrões envolvidos, e por essa razão, sugerem o uso de combinações do tipo *p* para *p-1*. Por outro lado, uma vez que algumas combinações são casos especiais de outras (combinações 2 para 1 são casos especiais das 3 para 2, por exemplo), ao começar com combinações de índices mais elevados (4 para 3, por exemplo), os autores acabam por evitar todos os outros casos de índices mais baixos (2 para 1, por exemplo).

Foerster e Wascher descrevem experiências computacionais em instâncias com um máximo de 40 tamanhos de itens diferentes. As soluções iniciais com base nas quais as heurísticas KOMBI são testadas são geradas a partir de uma heurística para o problema de corte standard a uma dimensão. Á medida que a dimensão das instâncias vai ficando maior, os tempos de execução das heurísticas que foram testadas também aumenta. O caso mais difícil que os autores abordam é o das instâncias em que a procura média por item é de 100 unidades, o número de itens diferentes é igual a 40, e o tamanho do item mais pequeno pode chegar a 1% do tamanho do rolo. Os autores testaram algumas heurísticas KOMBI em 100 instâncias desse tipo, obtendo tempos médios entre os 184 segundos e os 379 segundos. Os autores comparam as heurísticas KOMBI a abordagem de Diegel *et al.* [D93], e concluem que obtêm resultados ligeiramente piores apenas em instâncias pequenas em que os itens são grandes e a procura média é elevada.

2.3.3 Heurística de Yanasse e Limeira

Em [Y06], Yanasse e Limeira descrevem uma heurística híbrida para resolver o Problema de Minimização de Padrões baseada naquilo que os autores designam por "bons padrões". Um bom padrão consiste num padrão com um desperdício reduzido, e a partir do qual é possível satisfazer totalmente a procura de pelo menos dois itens. Esses padrões incluem forçosamente itens cuja procura é igual ou múltipla.

Os autores começam por gerar esses padrões usando a *Repeated Pattern Exhaustion Technique* (RPET). Para ser incluído na solução, um padrão gerado por essa via terá também de satisfazer um determinado nível de aspiração, que definiremos mais adiante. Quando a heurística não consegue encontrar mais nenhum bom padrão, os autores resolvem o problema

de corte a uma dimensão associado à instância residual, e aplicam técnicas de redução de padrões como aquelas que analisámos na Secção anterior.

O nível de aspiração que define as restrições a que é sujeito um padrão que irá fazer parte da solução é composto pelos seguintes parâmetros:

i) Um nível de utilização G;
ii) Uma eficiência mínima β;
iii) O facto de a procura de mais do que um item ter de ser satisfeita quando o padrão é usado G vezes.

Em relação ao parâmetro G, os autores usam submútliplos das procuras, uma vez que procuram padrões que possam satisfazer a procura de vários itens ao mesmo tempo. Começam pelo maior desses submúltiplos, e vão testando cada um desses valores por orderm decrescente.

Para o valor de β, os autores consideram duas situações para a aplicação desse parâmetro. Numa fase inicial do processo de geração dos padrões, os autores fazem $\beta=1-\alpha_s$, em que α_s corresponde ao valor médio do desperdício da instância residual que é calculado através da resolução do problema de corte standard a uma dimensão. Numa segunda fase, os autores impõem que os padrões devam ter pelo menos um desperdício igual a $\beta=1-0.8\alpha_s$. Segundo os autores, os resultados são sensíveis a esses valores de β.

Yanasse e Limeira testaram essa heurística num conjunto de 1800 instâncias divididas em 18 classes de 100 instâncias cada. O número de itens diferentes varia entre os 10 e os 40, e as procuras médias por itens entre as 10 e as 100 unidades. O tamanho dos itens é também um dos parâmetros das instâncias. No pior caso, o item mais pequeno poderá ter um tamanho de apenas 1% do rolo. No outro extremo, o tamanho do item mais pequeno é pelo menos igual a 20% do tamanho do rolo. Nas experiências computacionais que descrevem, Yanasse e Limeira mostram que é possível obter soluções melhores do que aquelas que podem ser obtidas com outras heurísticas usando a heurística híbrida proposta por eles. Os tempos de execução são também baixos, uma vez que no pior caso são necessários apenas 50.61 segundos (em média) para se calcular essas soluções.

2.4 Conclusões

Como vimos neste Capítulo, o número de algoritmos exactos propostos para o PMP é muito reduzido. Por outro lado, actualmente, os melhores algoritmos só permitem resolver algumas instâncias de média e pequena dimensão. O modelo que iremos abordar no próximo Capítulo e que foi introduzido complementa os resultados alcançados até agora nesse domínio. Esse modelo serviu de base para os trabalhos realizados ao longo desta dissertação. A constatação de que muito poucas abordagens de resolução exacta para o Problema de Minimização de Padrões tinham sido propostas até agora é outra das motivações deste trabalho.

A literatura descreve essencialmente heurísticas para o problema. Essas heurísticas baseiam-se na tentativa de identificar/gerar padrões cujos níveis de utilização sejam elevados. A primeira heurística foi proposta já em 1975 por Haessler [H75]. Nos últimos anos, foram propostas heurísticas que procuram combinar padrões de modo a reduzir o número de padrões distintos. A abordagem mais recente que analisámos na última Secção tira partido dessas combinações, e também de algumas ideias contidas nos trabalhos de Haessler [H75] ao nível da geração de padrões com altos níveis de utilização.

Capítulo 3

Um Novo Modelo de Geração de Colunas

3.1 Introdução

Num artigo muito recente ([AMVC09]), Alves *et al.* descrevem um modelo de geração de colunas para o PMP baseado numa nova decomposição de Dantzig-Wolfe. Do ponto de vista teórico, pode-se mostar que o modelo proposto por esses autores é mais forte que os modelos propostos até agora. Os autores compararam o valor da relaxação linear do seu modelo com os melhores resultados da literatura, nomeadamente com o modelo de geração de colunas de Vanderbeck [V00] e com as melhorias propostas por Alves e Carvalho [AVC08].

Em [AMVC09], os autores descrevem várias estratégias para reforçar o modelo. Descrevem novas restrições para o modelo e definem limites nos coeficientes do problema mestre. Esses coeficientes são variáveis dos subproblemas. Uma das abordagens usadas por Alves *et al.* baseia-se na técnica de Programação por Restrições. Os autores mostram também nesse artigo que é possível calcular limites inferiores de forma muito eficiente usando um modelo de Programação por Restrições para o PMP. Esses modelos permitem tratar as restrições complexas do modelo, nomeadamente as restrições não-lineares do PMP.

Neste Capítulo, revemos esse modelo em detalhe uma vez que os resultados desta dissertação se baseiam nele. Apresentamos também alguns procedimentos novos para reforçar esse modelo. Em [AMVC09], os autores usaram o modelo para calcular exclusivamente limites inferiores para o PMP. Nesta dissertação, usamos pela primeira vez o

modelo para calcular soluções inteiras para o PMP. As estratégias seguidas são descritas no próximo Capítulo.

3.2 O Modelo

O modelo desenvolvido por Alves *et al.* em [AMVC09] corresponde a uma decomposição de Dantzig-Wolfe do modelo compacto e não linear (1)-(8) no qual apenas as restrições de procura são mantidas no problema mestre. Nessa decomposição, existem agora dois subproblemas: o problema de mochila que permite definir os padrões válidos de corte, e um novo subproblema em que se definem as multiplicidades dos padrões que serão usados. Esses subproblemas são definidos respectivamente através das restrições (3) e (4)-(5) do modelo não-linear de Vanderbeck.

O modelo é o seguinte:

$$\min z_{AMVC} = \sum_{p \in P} d_p \delta_p \tag{24}$$

sujeito a

$$\sum_{p \in P} a_{np}^{mult} \delta_p - \sum_{k \in K, n \leq u_k} \lambda_{kn} = 0 \qquad n = 1, \dots, n^{max}, \tag{25}$$

$$\sum_{k \in K} \sum_{n=1}^{u_k} n a_{ikn}^{items} \lambda_{kn} = b_i, \qquad i = 1, \dots, m, \tag{26}$$

$$\sum_{p \in P} \delta_p = 1 \tag{27}$$

$$\lambda_{kn} \in \{0,1\}, \qquad k \in K, n = 1, \dots, u_k, \tag{28}$$

$$\delta_p \in \{0,1\}, \qquad p \in P. \tag{29}$$

Existem dois conjuntos de variáveis. As variáveis δ_p estão associadas às combinações de multiplicidades. São variáveis binárias, e apenas uma dessas variáveis poderá ter o valor 1 (restrição (27)). As restantes serão todas iguais a 0. As colunas associadas a δ_p representam

combinações válidas de multiplicidades de padrões. Uma combinação p é válida se a soma das multiplicidades incluídas nessa combinação for inferior ou igual a z_{CSP}:

$$\sum_{n=1}^{n^{max}} n a_{np}^{mult} \leq z_{CSP}.$$

Obviamente, se z_{CSP} corresponder ao número mínimo de rolos obtido resolvendo o problema de corte standard a uma dimensão, a restrição corresponderá efectivamente a uma equação. Os coeficientes a_{np}^{mult} represesentam o número de vezes que uma combinação p inclui a multiplicidade n. Dado que o valor máximo de uma multiplicidade corresponde a n^{max}, para cada combinação são necessários coeficientes para as multiplicidades 1 até à n^{max}. O conjunto P representa o conjunto de todas as combinações possíveis de multiplicidades. A grande novidade em relação ao modelo de geração de colunas de Vanderbeck prende-se com esse aspecto do modelo. O modelo (24)-(29) permite identificar de forma clara as combinações de multiplicidades que são usadas numa solução, o que não acontece com o modelo de geração de colunas de Vanderbeck. Outra vantagem em termos numéricos é que se consegue contornar um dos inconvenientes do modelo de Vanderbeck que consiste na tendência das colunas com maiores multiplicidades a concentrar valores positivos muito pequenos ("*LP cheating*"). No modelo de geração de colunas de Vanderbeck, todas as colunas têm o mesmo custo o que explica essa tendência. Os coeficientes elevados nas restrições de procura fazem com que seja possível satisfazer uma grande fracção dessas procuras com base em pequenos valores das variáveis. Como os custos das colunas são todos iguais, o modelo não consegue penalizar esse fenómeno. No modelo (24)-(29), só as colunas que correspondem a combinações de multiplicidades têm um custo positivo. Esse custo corresponde ao número de padrões diferentes na combinação p, e é designado por d_p no modelo acima. A multiplicidade não aparece directamente no conjunto de coeficientes do lado esquerdo das restrições, evitando-se assim esse fenómeno de *LP cheating*.

As colunas associadas às variáveis λ_{kn} correspondem a padrões de corte válidos replicados n vezes (a multiplicidade do padrão), e têm um coeficiente -1 na linha n que permite associar a multiplicidade ao padrão. As restrições (25) exprimem o facto de se uma multiplicidade tiver sido escolhido α vezes, não poderá haver mais de α padrões de corte com essa multiplicidade. Finalmente, as restrições (26) são restrições típicas de procura.

O problema mestre é constituído por 2 grandes blocos conforme ilustrado na Figura 3.1. O primeiro bloco diz respeito às combinações de multiplicidades enquanto o segundo corresponde aos padrões de corte. O segundo bloco subdivide-se em n^{max} subconjuntos, um por cada multiplicidade.

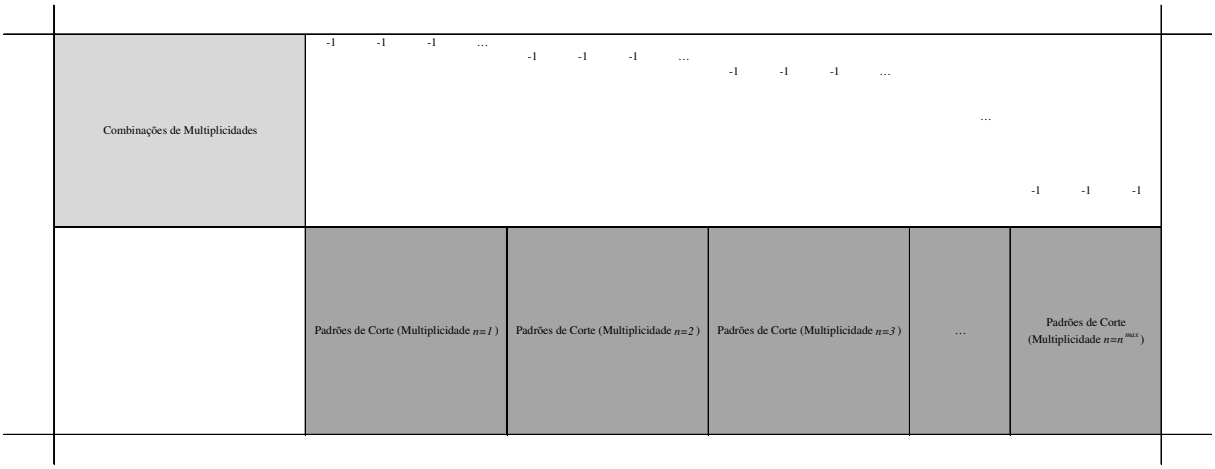

Figura 3.1: Estrutura angular em blocos do modelo (24)-(29)

A Figura 3.1 permite ganhar uma percepção mais intuitiva do modelo. A reserva das multiplicidades é feita no primeiro bloco. A segunda parte do modelo (o segundo bloco) permite fazer a distribuição das multiplicidades reservadas por vários padrões de corte, de modo a satisfazer a procura total dos itens. O Exemplo seguinte completa a exposição.

Exemplo 3.1

Vamos considerar novamente a pequena instância do PMP que usámos no Capítulo anterior. A Figura 3.2 ilustra esquematicamente a estrutura do modelo (24)-(29) para essa instância. Por razões de espaço, não fazemos a enumeração de todas as colunas do modelo. As colunas excluídas da Figura correspondem a padrões de corte válidos.

A solução óptima inteira é também indicada na Figura 3.2. O número mínimo de padrões diferentes é igual a 3. A solução corresponde a usar 3 multiplicidades (1, 3 e 4). Cada multiplicidade representa um rolo que é usado. No mínimo, os itens são cortados a partir de 8 rolos, o que corresponde exactamente à soma das multiplicidades usadas. A coluna que corresponde a essa combinação tem um custo associado igual a 3, precisamente o número de padrões diferentes que poderão ser escolhidos na segunda parte do modelo.

			Multiplicidades					Itens						
W=15			*n*					b_i	5	3	4	3	2	Função
z	**=**	**3**	1	2	3	4	5	w_i	9	7	5	4	2	Objectivo
	1	0			1		1							2
	2	0	1	1			1							3
	3	0				2								2
	4	**1**	1		1	1								3
	5	0		2		1								3
	6	0	2	1		1								4
	7	0	4			1								5
Combinações	8	0		1	2									3
de	9	0	2		2									4
multiplicidade	10	0	1	2	1									4
	11	0	3	1	1									5
	12	0	5		1									6
	13	0		4										4
	14	0	2	3										5
	15	0	4	2										6
	16	0	6	1										7
	17	0	8											8
	18	0	-1					0	0	0	0	1		13
	19	0	-1					0	0	0	0	2		11
	20	0	-1					0	0	0	1	0		11
	21	0	-1					0	0	0	1	1		9
	22	0	-1					0	0	0	1	2		7
	23	0	-1					0	0	0	2	0		7
	24	0	-1					0	0	0	2	1		5
	25	0	-1					0	0	0	2	2		3
	26	0	-1					0	0	0	3	0		3
	27	0	-1					0	0	0	3	1		1
	39	0	-1					0	0	2	1	0		1
	40	0	-1					0	0	3	0	0		0
	41	0	-1					0	1	0	0	0		8
	45	0	-1					0	1	0	1	1		2
	46	0	-1					0	1	0	1	2		0
	47	0	-1					0	1	0	2	0		0
	48	0	-1					0	1	1	0	0		3
	49	0	-1					0	1	1	0	1		1
	50	0	-1					0	2	0	0	0		1
	51	0	-1					1	0	0	0	0		6
	52	0	-1					1	0	0	0	1		4
	53	**1**	-1					1	0	0	0	2		2
	54	0	-1					1	0	0	1	0		2
	55	0	-1					1	0	0	1	1		0
	56	0	-1					1	0	1	0	0		1
	57	0		-1				0	0	0	0	2		26
Padrões de corte	59	0		-1				0	0	0	2	0		22
válidos	60	0		-1				0	0	0	2	2		18
	67	0		-1				0	0	2	0	0		20
	68	0		-1				0	0	2	0	2		16
	70	0		-1				0	0	2	2	0		12
	71	0		-1				0	0	2	2	2		8
	75	0		-1				0	0	4	0	0		10
	76	0		-1				0	0	4	0	2		6
	84	0		-1				0	2	0	2	2		4
	87	0		-1				0	2	2	0	0		6
	88	0		-1				0	2	2	0	2		2
	90	0		-1				2	0	0	0	0		12
	91	0		-1				2	0	0	0	2		8
	93	0		-1				2	0	0	2	0		4
	94	0		-1				2	0	0	2	2		0
	95	0		-1				2	0	2	0	0		2
	106	0			-1			0	0	3	0	0		30
	109	0			-1			0	0	3	3	0		18
	119	0			-1			0	3	0	0	0		24
	122	**1**			-1			0	3	0	3	0		12
	126	0			-1			0	3	3	0	0		9
	129	0			-1			3	0	0	0	0		18
	132	0			-1			3	0	0	3	0		6
	134	0			-1			3	0	3	0	0		3
	145	0				-1		0	0	4	0	0		40
	168	0				-1		4	0	0	0	0		24
	173	**1**				-1		4	0	4	0	0		4
	207	0					-1	5	0	0	0	0		30

Desperdicio

Figura 3.2: Representação do modelo (24)-(29) para a instância do Exemplo 3.1

Na parte inferior da Figura 3.2 encontram-se os padrões de corte. Todas as variáveis associadas a esses padrões têm um custo associado igual a 0. Na Figura 3.2, a coluna *Desperdício* é meramente informativa. Esses dados não fazem parte do modelo (24)-(29). Os padrões de corte escolhidos correspondem respectivamente às combinações de itens (tamanhos) e multiplicidades seguintes:

Padrão 1: 9 + 2 + 2 (Multiplicidade = 1) (Desperdício = 1 × 2 = 2)
Padrão 2: 7 + 4 (Multiplicidade = 3) (Desperdício = 3 × 4 = 12)
Padrão 3: 9 + 5 (Multiplicidade = 4) (Desperdício = 4 × 1 = 4)
(Desperdício total: 18)

Cada um desses padrões é usado uma vez, dado que as variáveis associadas são variáveis binárias.

Existe apenas um padrão de multiplicidade igual a 5, tal como acontecia no modelo de geração de colunas de Vanderbeck. Isso deve-se ao facto de existir apenas um item com procura igual a 5. Todos os outros itens têm procuras inferiores a 5, e não podem por isso estar em padrões com essa multiplicidade uma vez que a procura dos itens deve ser satisfeita exactamente. Consequentemente, o número de colunas em cada subconjunto no segundo bloco do modelo tende a decrescer à medida que as multiplicidades vão aumentando. O subproblema de geração de colunas associado a cada um desses subconjuntos tende ele também a ser cada vez mais fácil para valores crescentes das multiplicidades.

É fácil observar que algumas combinações de multiplicidades nunca serão usadas na solução óptima inteira. A combinação de multiplicidades "4 + 4" é um exemplo. Sabendo que existem itens cuja procura é estritamente inferior a 4, sabemos à partida que serão necessários padrões de multiplicidade inferior a 4. É o caso dos itens de tamanho 4 e 2 cujas procuras são respectivamente iguais a 3 e 2. Em [AMVC09], os autores excluem essas combinações impondo restrições directamente no subproblema de geração de colunas. Os autores propõem também outras formas de reforçar o modelo conforme veremos mais adiante.

Na Figura 3.2, não representámos a restrição (27) do modelo de Alves *et al*. Essa restrição garante que é usada exactamente uma combinação de multiplicidades entre todas as combinações existentes. ■

3.3 Os subproblemas de Geração de Colunas

Dado o número muito elevado de colunas em cada um dos dois blocos do modelo (24)-(29), para qualquer instância de média ou grande dimensão, é necessário recorrer à geração diferida de colunas: o problema mestre deve ser inicializado com um número restrito de colunas e resolvido como um problema de Programação Linear normal; a solução dual que é obtida no final dessa resolução permite identificar as colunas que não foram enumeradas e que poderão melhorar o valor da função objectivo (colunas com custo reduzido negativo). No caso do modelo (24)-(29), é necessário resolver dois subproblemas distintos para poder obter o óptimo da relaxação linear.

O primeiro subproblema diz respeito ao primeiro bloco do modelo, e consiste em determinar a combinação de multiplicidades mais atractiva. Vamos designar por π o vector de variáveis duais associadas às restrições (25) e por β a variável dual associada à restrição (27). O custo reduzido de uma coluna p no primeiro bloco do modelo é igual a

$$\sum_{n=1}^{n^{max}} a_{np}^{mult} - \sum_{n=1}^{n^{max}} \pi_n a_{np}^{mult} - \beta$$

A quantidade $\sum_{n=1}^{n^{max}} a_{np}^{mult}$ representa o custo de uma coluna, e está directamente associadoao número de padrões diferentes de uma solução. Dado que β é uma constante, podemos omiti-la no problema de optimização. Designando por x_n^{mult} as variáveis do subproblema, podemos formular esse problema da forma seguinte:

$$\min z_{SP1} = \sum_{n=1}^{n^{max}} (1 - \pi_n) x_n^{mult} \tag{30}$$

sujeito a

$$\sum_{n=1}^{n^{max}} n x_n^{mult} \leq z_{CSP} \tag{31}$$

$$x_n^{mult} \in \mathbb{N}, \qquad n = 1, \dots, n^{max}. \tag{32}$$

Conforme se pode observar, o problema (30)-(32) é um problema de mochila clássico em que o custo dos itens é igual a $1 - \pi_n$. Esse problema pode ser resolvido de forma eficiente usando por exemplo algoritmos de Programação Dinâmica. Em [AMVC09], os autores impõem restrições adicionais nesse subproblema. Algumas dessas restrições são limites superiores para as variáveis x_n^{mult}, e não alteram a complexidade do problema. Outras restrições impostas pelos autores como por exemplo o número mínimo de multiplicidades a considerar têm um impacto directo na estrutura do problema. Essa última restrição pode ser formulada desta forma:

$$\sum_{n=1}^{n^{max}} n x_n^{mult} \geq z_{CSP}, \tag{33}$$

em que LB_{PMP} representa um limite inferior para o número de padrões diferentes (a solução do PMP). Essa restrição permite reduzir o número de colunas no problema mestre, e assim reforçá-lo, mas complica também o problema que deixa de ser um problema simples de mochila. Se for conhecido um limite superior para o PMP, uma restrição semelhante à (33) poderá também ser imposta.

O segundo subproblema permite identificar os padrões de corte válidos que poderão melhorar a função objectivo global. O custo reduzido de uma coluna que corresponde ao padrão *k* de multiplicidade *n* é igual a

$$\pi_n - \sum_{i=1}^{m} \theta_i n a_{ikn}^{items}.$$

Dado que π_n é constante, minimizar esse custo reduzido é equivalente a maximizar o segundo termo da expressão. Se fixarmos a multiplicidade *n*, e dado que as restrições dizem apenas respeito ao comprimento do rolo e que as procuras dos itens devem ser satisfeitas exactamente, obtemos um problema de mochila clássico. Na realidade este segundo

subproblema é em tudo igual ao subproblema de geração de colunas usado por Vanderbeck em [V00]. Inicialmente, o problema é não-linear (o parâmetro *n* na expressão do custo reduzido é uma variável), mas é facilmente linearizado fixando o valor da multiplicidade. Tal como em [V00], não é necessário resolver um subproblema por cada valor sucessivo da multiplicidade uma vez que uma solução óptima se pode manter óptima para várias multiplicidades sucessivas.

3.4 Estratégias de Reforço

Alves *et al.*propõem diferentes métodos para refoçar o modelo (24)-(29). As estratégias de reforço passam por impor restrições às colunas associadas às combinações de multiplicidades. Os autores usam também um modelo de Programação por Restrições para derivar limites superiores relativamente ao número máximo de vezes que uma multiplicidade pode ser considerada. Finalmente, os autores descrevem uma família de cortes que podem ser adicionados ao problema mestre.

Uma das restrições mais fortes descrita em [AMVC09] baseia-se no facto de um item com uma procura b_i só poder aparecer em padrões cuja multiplicidade seja inferior ou igual a b_i. Se derivarmos um limite inferior r_i relativamente ao número de rolos que são necessários para cortar todos os itens cujas procuras sejam inferior ou igual a b_i, poderemos afirmar que uma combinação *p* de multiplicidades só será válida se

$$\sum_{n=1}^{b_i} n a_{np}^{mult} \geq r_i.$$

A título de exemplo, vamos considerar a instância do Exemplo 3.1, e em particular o caso dos itens com procura inferior ou igual a 4. Para cortar esses itens, são necessários pelo menos 4 rolos:

$$\left\lceil \frac{2 \times 2 + 4 \times 3 + 5 \times 4 + 7 \times 3}{15} \right\rceil = 4.$$

A primeira combinação de multiplicidades descrita na Figura 3.2 corresponde a usar uma multiplicidade de valor igual a 3 e outra de valor igual a 5, o que significa que até à

multiplicidade 4, só poderão ser usados 3 rolos. Essa combinação não é por isso válida e pode ser excluída do conjunto *P*.

O segundo conjunto de restrições é similar ao anterior mas diz respeito agora ao número mínimo de padrões que devem ser usados até uma dada multiplicidade. Se soubermos que não é possível satisfazer a procura dos itens com procura inferior ou igual a b_i com menos de p_i padrões, então a restrição que segue é válida

$$\sum_{n=1}^{b_i} a_{np}^{mult} \geq p_i.$$

O limite inferior p_i pode ser calculado por exemplo resolvendo um problema de empacotamento em que a procura dos itens é igual a 1.

Alves *et al.* derivaram diferentes limites superiores para o valor dos a_{np}^{mult}. Uma das estratégias que é seguida pelos autores baseia-se no paradigma de Programação por Restrições. Os autores usam um modelo desse tipo para representar o PMP. Os limites superiores são calculados resolvendo problemas de satisfação de restrições em que se impõem restrições do tipo $a_{np}^{mult} \geq h$. Se o modelo não tiver nenhuma solução válida, então *h-1* é necessariamente um limite superior para a_{np}^{mult}. Na Secção seguinte, descrevemos em detalhe o modelo de Programação por Restrições.

O modelo (24)-(29) também pode ser reforçado recorrendo a planos de corte que são inseridos directamente na relaxação linear do modelo. Os cortes descritos em [AMVC09] baseiam-se nos itens cuja procura é ímpar. Atendendo às restrições de procura (26), sempre que um item tem uma procura ímpar, na solução deverá sempre haver pelo menos uma coluna cujo coeficiente na restrição de procura do respectivo item seja também ímpar. O corte associado corresponde à seguinte restrição:

$$\sum_{k \in C}^{n^{max}} \lambda_{kn} \geq 1\,, \qquad \forall i\colon b_i \text{ é ímpar},$$

em que *C* representa o conjunto de padrões tal que $n \times a_{ikn}^{items}$ é ímpar. Numa solução inteira, essa restrição é obviamente satisfeita. Numa solução fraccionária, o mesmo não acontece necesariamente, e itens com procura ímpares poderão ser satisfeitos apenas com base em coeficientes pares. É importante salientar que esses cortes são válidos apenas porque as procuras devem ser satisfeitas exactamente. Se fosse permitido produzir mais unidades de um dado item, as procuras ímpares poderiam ser satisfeitas com mais unidades, eventualmente a partir de um número par de itens.

3.5 Um Modelo de Programação por Restrições

Os modelos de Programação por Restrições podem ser usados para exprimir de forma eficiente restrições complexas que são difíceis de tratar com modelos de Programação Inteira. Alves *et al.* usaram um modelo desse tipo para poder ter em conta a restrição não-linear (10) do modelo compacto (9)-(12). Os autores usaram esse modelo restringir o conjunto de combinações de multiplicidades válidas do modelo (24)-(29), e também para calcular limites inferiores para o PMP.

Muitas combinações de multiplicidades não são interessantes, pois nunca irão fazer parte da solução óptima inteira. Uma combinação de multiplicidades só é válida se permitir recuperar todas as procuras dos itens. Dito de outra forma, deve ser possível exprimir todas as procuras como uma combinação linear das multiplicidades escolhidas. A título de exemplo, vamos recorrer novamente à instância do Exemplo 3.1. A combinação de multiplicidades cujo índice é 5 corresponde a usar as multiplicidades 2, 2 e 4. Não é possível com base nessa combinação exprimir as procuras ímpares 3 e 5. Não existe nenhuma combinação linear das multiplicidades 2, 2 e 4 que permita recuperar o valor 3 ou 5. Logo essa combinação não é válida, e idealmente deveria ser excluída.

Em [AMVC09], os autores formularam um modelo de Programação por Restrições que toma em conta explicitamente essa restrição. O modelo usa variáveis X_j que representam o valor da $j^{ésima}$ multiplicidade. As multiplicidades estão ordenadas por ordem crescente através das seguintes restrições

$$X_j \leq X_{j+1} \vee X_{j+1} = 0, \;\; j = 1, \dots, ub_{PMP} - 1, \tag{34}$$

$$X_j = 0 \Rightarrow X_{j+1} = 0, \;\; j = 1, \dots, ub_{PMP} - 1, \tag{35}$$

com ub_{PMP} a representar um limite superior no número de padrões diferentes, e consequentemente no número de multiplicidades que podem ser escolhidas. A restrição (34) define a ordenação das multiplicidades. Se forem usadas menos multiplicidades do que o limite ub_{PMP}, então as últimas multiplicidades da combinação serão postas 0 (essa é a única violação à regra de ordenação). Dessa multiplicidade em diante, todas as multiplicidades deverão ser iguais a 0 (restrição (35)).

O modelo de Programação por Restrições define explicitamente as restições que obrigam a que cada procura possa ser expressa como uma combinação linear das multiplicidades:

$$\sum_{j=1}^{ub_{PMP}} A_{ij} X_j = b_i, \;\; i = 1, \dots, m' \leq m. \tag{36}$$

Essas restrições correspondem às restrições não-lineares de procura do modelo compacto (9)-(12). Contudo, neste caso, apenas importa o valor das procuras, e não o item propriamente dito. Nem o item, nem o seu tamanho são considerados explicitamente, o que permite fazer várias simplificações. Desde já, se tivermos dois itens com a mesma procura, haverá apenas uma restrição do tipo (36) para ambos os itens. Outras simplificações são descritas em [AMVC09].

Os limites superiores quanto ao número de vezes que uma multiplicidade pode ser usada é formulada através das restrições seguintes:

$$count(X, n) \leq ub\left(a_{np}^{mult}\right), \qquad n = 1, \dots, n^{max},$$

sendo *X* o vector de variáveis do modelo e *count* uma função de contagem. Uma multiplicidade representa também um número de rolos que é usado, e como tal a soma de todas as multiplicidades não poderá nunca exceder o limite de z_{CSP} rolos:

$$\sum_{j=1}^{ub_{PMP}} X_j \leq z_{CSP}.$$

Finalmente, os cortes que introduzimos na Secção anterior também são considerados neste modelo.

Como já dissemos, Alves *et al.* usaram esse modelo para derivar limites superiores sobre o número de vezes que uma multiplicidade podia ser incluída numa solução, mas também o usaram para derivar limites inferiores para o PMP. O procedimento que seguiram para esse efeito é simples: começam por impor uma restrição do tipo

$$X_{lb_{PMP}+1} = 0$$

e resolverm o problema de satisfação de restrições aaaociado. Se problema não tiver solução, o limite inferior é actualizado passando para $lb_{PMP} + 1$. O processo repete-se até que o problema tenha uma solução válida. Quando isso acontece, conclui-se que o limite inferior é igual a lb_{PMP}.

Os autores testaram o seu modelo de Programação por Restrições nas instâncias reais de Vanderbeck [V00], e obtiveram em alguns casos limites inferiores de melhor qualidade do que aqueles que é possível obter com o modelo de geração de colunas de Vanderbeck. Em qualquer dos casos, o limite inferior é calculado em tempos computacionais muito reduzidos. Em média, os tempos são 8 vezes menores que os tempos necesários à resolução do modelo de Vanderbeck.

3.6 Novas Soluções de Reforço

Nesta Secção descrevemos estratégias complementares que poderão ser usadas para reforçar o modelo (24)-(29). Em [AVC08], os autores usam funções duais válidas para determinar inequações válidas, e assim reforçar a relaxação linear do problema mestre. Essas funções podem também ser usadas no caso do modelo (24)-(29). Os autores defendem também que é possível obter melhores resultados se em vez das restrições originais do modelo se usarem

combinações de restrições. Uma combinação será tanto melhor quanto melhor for a relação entre os coeficientes do lado esquerdo e o termo do lado direito. O Exemplo seguinte ilustra essa observação.

Exemplo 3.2

A título ilustrativo, vamos usar a função dual válida $u^{(k}(x)$ introduzida em [FS01]:

$$u^{(k)}(x)\colon [0,1] \to [0,1]$$

$$u^{(k)}(x) = \begin{cases} x, & se\ (k+1)x \in \mathbb{Z}, \\ \dfrac{\lfloor (k+1)x \rfloor}{k}, & caso\ contrário. \end{cases}$$

Para simplificar a apresentação, vamos conisderar o caso em que $k=1$. Vamos também assumir que as duas restrições seguintes são válidas para um determinado modelo:

$$5{,}6x_1 + 8x_2 = 12 \tag{37}$$

$$x_2 \geq 1 \tag{38}$$

Normalizando a primeira restrição, obtemos a equação seguinte:

$$\frac{5{,}6}{12}x_1 + \frac{8}{12}x_2 = 1.$$

Se aplicarmos a função para $k=1$ à equação acima, obtemos a seguinte inequação válida:

$$x_2 \leq 1 \tag{39}$$

Se combinarmos a restrição (37) e (38), subtraindo a segunda à primeira, obtemos a seguinte restrição:

$$5{,}6x_1 + 7x_2 \leq 11. \tag{40}$$

Aplicando a função dual válida a (40), obtemos a inequação válida seguinte:

$$x_1 + x_2 \leq 1, \tag{41}$$

que é mais forte que (40). ■

Para cada item *i, i=1,…,m,* é possível calcular um limite inferior para o número de rolos necessários para os cortar. Vamos designar por z_i o valor desse limite. A seguinte família de restrições é válida para o modelo (24)-(29):

$$\sum_{k \in K} \sum_{n=1;\, a_{ikn}^{items}>0}^{u_k} n\lambda_{kn} \geq z_i, \qquad i = 1, \dots, m. \tag{42}$$

A restrição (42) determina que a soma dos rolos que incluem o item *i* deve ser pelo menos igual ao limite inferior z_i. Se combinarmos (42) com as restrições de procura (26) do modelo (24)-(29), obtemos as seguintes restrições:

$$\sum_{k \in K} \sum_{n=1}^{u_k} max\{0, na_{ikn}^{items} - n\}\lambda_{kn} \leq b_i - z_i, \qquad i = 1, \dots, m, \tag{43}$$

O coeficiente $max\{0, na_{ikn}^{items} - n\}$ impede que haja coeficientes negativos na equação do corte. Se um padrão *k* incluir o item *i* $(a_{ikn}^{items} > 0)$, então $na_{ikn}^{items} - n$ é necessariamente superior ou igual a 0.

Em certos casos, é de esperar que a relação entre os coeficientes do termo do lado esquerdo na restrição (43) seja mais favorável do que na restrição de procura original (26). Para esses casos, os cortes que poderão ser derivados a partir de funções duais válidas aplicadas às restrições (43) serão potencialmente mais fortes.

Outra forma de reforçar o modelo (24)-(29) consiste em impor uma restrição que obriga a que haja pelo menos uma multiplicidade ímpar se z_{CSP} for ímpar e corresponder ao número mínimo de rolos. Essa restrição pode ser imposta directamente no subproblema de geração de colunas com o qual se geram as combinações de multiplicidades. Em alternativa, podemos impor a restrição seguinte no problema mestre:

$$\sum_{k \in K} \sum_{n=1;\, n\ \acute{i}mpar}^{u_k} \lambda_{kn} \quad \geq 1.$$

Essa restrição é útil uma vez que Alves *et al.* não garantem que todas as combinações não válidas de multiplicidades são excluídas com as estratégias descritas em [AMVC09].

3.7 Conclusões

Neste Capítulo, descrevemos um novo modelo de geração de colunas proposto em [AMVC09]. Esse modelo é teoricamente mais forte que o modelo de geração de colunas de Vanderbeck [V00]. Os resultados desta dissertação baseiam-se nesse modelo. Alguns dos resultados descritos em [AMVC09] foram revistos. Descrevemos o modelo de Programação por Restrições que Alves *et al.* usam para calcular limites inferiores para o problema, e também reforçar o modelo de geração de colunas.

Foram também propostas novas formas de reforçar esse novo modelo de geração de colunas. A primeira estratégia consistiu em combinar restrições válidas do modelo para melhorar a relação entre coeficientes e assim melhorar a qualidade dos cortes que é possível obter com base em funções duais válidas. Finalmente, propomos uma nova inequação válida para o problema que pode ser imposta no problema mestre ou num dos subproblemas de geração de colunas.

Capítulo 4

Algoritmos para o Cálculo de Soluções Inteiras

4.1 Introdução

Neste Capítulo, usamos o modelo descrito no Capítulo anterior para desenvolver algoritmos de partição e avaliação sucessivas que permitam obter soluções inteiras. Os algoritmos consistem em usar o método de partição e avaliação sucessivas baseado na relaxação linear do modelo (24)-(29). Dado que se trata de um modelo de geração de colunas, o algoritmo final será um algoritmo de partição e geração de colunas (*branch-and-price* na terminologia anglo-saxónica). Várias regras de partição são descritas. No Capítulo seguinte, essas regras são testadas usando instâncias da literatura.

4.2 Algoritmo de Partição e Geração de Colunas: Esquemas de Partição

O modelo (24)-(29) permite definir diferentes regras de partição. No nosso caso, uma regra de partição consiste numa ou mais restrições que são adicionadas a um problema de Programação Linear de forma a excluir uma solução fraccionária, mas nunca as soluções inteiras. Todas as regras de partição que dependem das variáveis associadas aos padrões de corte são derivadas com base no modelo de fluxos para o PMP. Essa estratégia foi proposta

originalmente em [AVC08]. As novidades prendem-se essencialmente com as restrições de partição que derivamos com base nas variáveis do primeiro bloco do modelo (24)-(29).

4.2.1 Regra de Partição Baseada em Variáveis de Fluxo: Esquema I

A primeira regra de partição que propomos é similar àquela que é usada em [AVC08]. O valor das variáveis nas colunas do segundo bloco do modelo (24)-(29) podem ser convertidos em fluxos num modelo de fluxos para o PMP semelhante ao que foi descrito no Capítulo 2. Cada coluna do segundo bloco do modelo representa um padrão de corte válido que pode ser convertido numa sequência de fluxos x_{ij}^n. Se a solução da relaxação do modelo (24)-(29) for fraccionária, existirá pelo menos uma variável x_{ij}^n também fraccionária. Se, pelo contrário, todas as variáveis x_{ij}^n que resultam da conversão forem inteiras, então será possível recuperar uma solução inteira para (24)-(29) sem recorrer ao método de partição e avaliação sucessivas.

Para convertermos a solução da relaxação linear do modelo (24)-(29) num conjunto de fluxos em arcos, assumimos que os itens nos padrões de corte estão ordenados por ordem decrescente do seu tamanho. Os itens são colocados a partir do início do rolo. As colunas cujo valor é positivo na relaxação linear convertem-se numa sequência de fluxos em arcos de valor igual ao valor da variável associada. Uma vez convertida a solução, escolhe-se o arco com valor fraccionário cuja multiplicidade é a maior e que corresponde ao maior item colocado o mais à esquerda. Vamos assumir que esse arco é o arco (*r,s*) relativo ao item de tamanho *s*-*r*, que a sua multiplicidade é igual a *n* e que o valor do respectivo fluxo é igual a f_{rs}^n. As restrições de partição que são impostas são as seguintes:

$$x_{rs}^n \leq \lfloor f_{rs}^n \rfloor, \qquad\qquad x_{rs}^n \geq \lceil f_{rs}^n \rceil.$$

A ideia é definir rapidamente o valor dos arcos que têm maior impacto na solução, *ie* aqueles com maior multiplicidade (que consomem rapidamente a "disponibilidade" em rolos z_{CSP}) e de tamanho elevado (que consomem rapidamente o tamanho dos rolos *W*). O Exemplo seguinte ilustra essa regra de partição.

Exemplo 4.1

Vamos considerar a instância real do PMP designada por kT03 e que é usada por Vanderbeck em [V00]. A instância é composta por 7 itens diferentes, e tem rolos de comprimento W=445. Os itens da instância e as respectivas procuras são os seguintes:

i	w_i	b_i
1	243	12
2	235	12
3	228	10
4	197	48
5	180	18
6	170	31
7	50	6

O número mínimo de rolos que são necessários para cortar todos os itens é igual a 66. Ao resolver o problema de corte standard a uma dimensão, o número de padrões diferentes foi igual a 6 (limite superior para o PMP). A multiplicidade máxima que um padrão de corte poderá ter é igual a 31.

A resolução da relaxação linear do modelo (24)-(29) deu a solução fraccionária que é descrita na Figura 4.1. O valor das variáveis é indicado no topo de cada coluna. A solução é fraccionária e tem um valor igual a 5. As combinações de multiplicidades estão a cinzento na Figura. As restantes colunas dizem respeito aos padrões de corte.

O padrão de corte relativo à primeira coluna é convertido num fluxo num único arco com origem no vértice 0 e destino no vértice 50:

$$x^1_{0,50} = 0{,}16.$$

O padrão associado à segunda coluna dá origem aos seguintes fluxos:

$$x^3_{0,228} = 0{,}23,$$
$$x^3_{228,278} = 0{,}23 \text{ e}$$
$$x^3_{278,328} = 0{,}23.$$

A conversão desses padrões em fluxos em arcos leva a uma solução em que o arco fraccionário com maior multiplicidade e que está associado ao maior item mais á esquerda é o arco (0,228) de multiplicidade 10 ($x^{10}_{0,228} = 0{,}93$).

As restrições de partição que seriam então adicionadas à relaxação linear seriam:

$$x^{10}_{0,228} \geq 1 \quad \text{e} \quad x^{10}_{0,228} \leq 0.$$

■

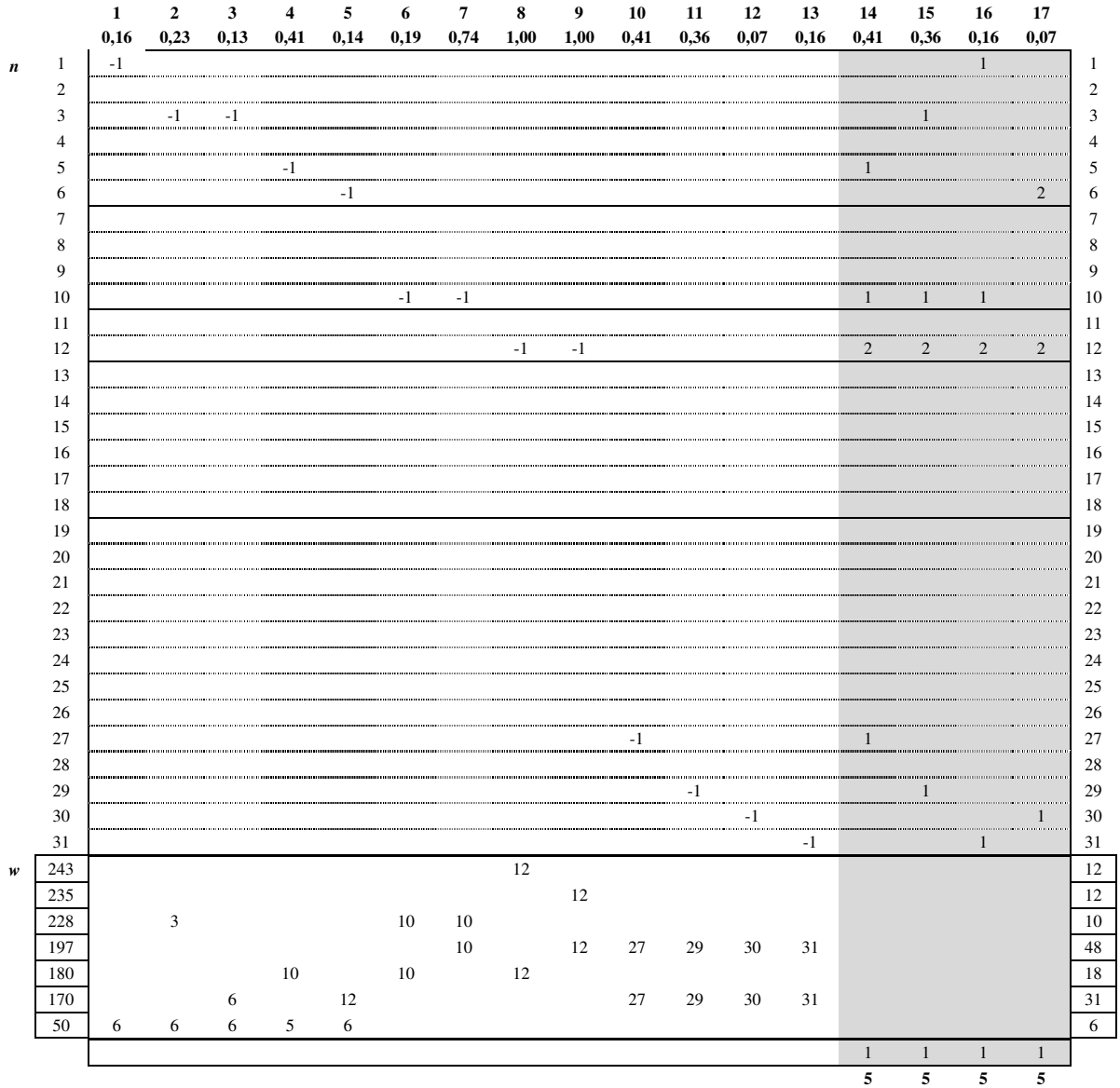

Figura 4.1: Solução da relaxação linear do problema mestre (Exemplo 4.1)

4.2.2 Regra de Partição Baseada em Variáveis de Fluxo: Esquema II

A regra de partição anterior impõe restrições fortes sobre itens simples num padrão com uma determinada multiplicidade. A restrição do tipo maior ou igual, por exemplo, obriga a que haja pelo menos $\lceil f_{rs}^n \rceil$ itens de tamanho *s*-*r* em padrões de multiplicidade *n.* Essa regra de partição pode conduzir a árvores que podem ser desequilibradas, *ie* em que o número de nodos num dos ramos é muito diferente do número de nodos no outro ramo. A segunda regra de partição que propomos tenta resolver esse problema.

Em vez de definir a partição a partir de uma variável de fluxo correspondente a um item colocado numa determinada posição de um padrão de multiplicidade *n*, vamos agora olhar para todos os padrões desde a multiplicidade 1 até à multiplicidade n^{max}, e impor as restrições de partição sobre as variáveis de fluxo agregado seguintes:

$$x_{rs} = \sum_{n=1}^{n^{max}} x_{rs}^n.$$

Na solução óptima da relaxação linear, se existir uma variável x_{rs} cujo valor f_{rs} é fraccionário, é efectuada uma partição baseada nas restrições seguintes:

$$x_{rs} \leq \lfloor f_{rs} \rfloor, \qquad\qquad x_{rs} \geq \lceil f_{rs} \rceil.$$

O facto de todas as variáveis x_{rs} serem inteiras não significa que a solução é inteira. Isso implica que essa restrição não possa ser usada só. Nas experiências computacionais que conduzímos a regra de partição descrita na Secção anterior é sempre usada em último recurso para garantir que o método converge para uma solução inteira.

Exemplo 4.2

Na instância kT03 descrita atrás e para solução fraccionária representada na Figura 4.1, a conversão em fluxos agregados daria o seguinte resultado para o arco (0,228):

$$x_{0,228} = x_{0,228}^3 + x_{0,228}^{10} = 0{,}23 + 0{,}93 = 1{,}16.$$

As restrições de partição que seriam impostas na sequência do esquema descrito nesta Secção seriam:

$$x_{0,228} \geq 2 \quad \text{e} \quad x_{0,228} \leq 1.$$

■

4.2.3 Regra de Partição Baseada em Variáveis de Fluxo: Esquema III

Finalmente, a última regra que exploramos e que se baseia em variáveis de fluxo tenta melhorar ainda mais o balanceamento da árvore de partição. A nossa terceira regra baseia-se não no fluxo num arco ou num conjunto de arcos associados a um mesmo item, mas sim no fluxo desde um determinado nodo do grafo. Indirectamente, essa regra condiciona o fluxo global num conjunto de arcos que são independentes das multiplicidades e dos itens.

Vamos designar por x_r o fluxo total desde um nodo *r* do grafo *G* que representa o PMP até ao nodo final *W*. O valor de x_r é dado pela equação seguinte:

$$x_r = \sum_{t \geq r;(t,s) \in A} x_{ts}^n.$$

Depois de resolvida a relaxação linear do modelo (24)-(29), a solução é convertida para um conjunto de fluxos conforme ilustrámos no Exemplo 4.1. Após essa conversão, é calculado o fluxo desde cada nodo do grafo até ao final, começando pelo nodo 0. Logo que se encontra um nodo cujo fluxo à sua direita é fraccionário (f_r) são criados dois nodos com base em restrições semelhantes áquelas que usámos anteriormente, nomeadamente:

$$x_r \leq \lfloor f_r \rfloor, \qquad x_r \geq \lceil f_r \rceil.$$

Novamente, esse esquema de partição não garante por si só que uma solução seja encontrada no final, pelo que teremos também de o combinar com outro.

4.2.4 Regra de Partição Baseada nas Variáveis do 1º Bloco: Esquema IV

Em alternativa às partições feitas com base em variáveis de fluxo, exploramos nesta dissertação outras regras baseadas nas novas variáveis introduzidas pelo modelo (24)-(29). Todas as regras que propomos são compatíveis com o subproblema de geração de colunas no qual se baseiam as colunas que correspondem a combinações de multiplicidades.

A primeira regra que propomos baseia-se no valor de uma única multiplicidade. O objectivo do primeiro bloco do modelo consiste em escolher uma única combinação de multiplicidades. Se na solução da relaxação linear do modelo houver mais de uma coluna desse bloco com valor positivo, isso significa obrigatoriamente que existem variáveis fraccionárias nesse bloco. Seja X_j^p o valor da $j^{ésima}$ multiplicidade da combinação p. Entre duas combinações p_1 e p_2 com valor positivo na relaxação linear, deve existir pelo menos um j tal que:

$$X_j^{p_1} \neq X_j^{p_2},$$

caso contrário as duas combinações seriam semelhantes. Em primeiro lugar, começamos por identificar a maior multiplicidade entre todas as combinações com valor positivo que não é comum a todas essas combinações. Seja X o valor dessa multiplicidade. Vamos também identificar o índice dessa multiplicidade em todas as combinações em que ocorre, e escolher o maior. Por exemplo, se a multiplicidade X só ocorrer em duas combinações (p_1 e p_2), e se em p_1 for a 5ª multiplicidade da combinação e se em p_2 ela for a 6ª multiplicidade da combinação, então registaremos o valor 6 como sendo o índice de referência da regra de partição que iremos impor. Uma vez seleccionada a multiplicidade (X) e o respectivo índice (j), criamos dois nodos a partir respectivamente das restrições de partição seguintes:

$$X_j^p \geq X, \quad \forall p \in P, \tag{44}$$

e

$$X_j^p \leq X - 1, \quad \forall p \in P. \tag{45}$$

A restrição (44) obriga a que a $j^{ésima}$ multiplicidade em qualquer padrão seja sempre pelo menos igual a *X*, enquanto a restrição (45) não permite que essa multiplicidade seja maior que *X-1*. Ao inserir várias restrições de partição desse género, a tendência irá no sentido de fixar uma determinada combinação de multiplicidades. Quando isso acontece, as variáveis do 1º bloco são todas inteiras atendendo à restrição do modelo (24)-(29) que obriga a que apenas uma combinação seja escolhida.

Escolher uma única combinação no primeiro bloco não significa que a solução do modelo seja ela também inteira. As variáveis associadas aos padrões de corte poderão ser fraccionárias. À semelhança das regras de partição que descrevemos nas secções anteriores, esta regra não é suficiente para atingir sistematicamente uma solução óptima inteira. É necessário por isso combiná-la com uma regra de partição complementar.

O Exemplo seguinte ilustra os detalhes deste esquema de partição.

Exemplo 4.4

Para a instância kT03 e a solução descrita na Figura 4.1, a maior multiplicidade que não é comum a todas as combinações de multiplicidades com valor positivo é a multiplicidade igual a 31. Essa multiplicidade é usada numa única combinação, e é a 5ª multiplicidade dessa combinação. As restrições de partição que devem ser impostas nesse caso são as seguintes:

$$X_5^p \geq 31, \quad \forall p \in P, \text{e}$$

$$X_5^p \leq 30, \quad \forall p \in P.$$

■

4.2.5 Regra de Partição Baseada nas Variáveis do 1º Bloco: Esquema V

Uma vez mais, a regra de partição anterior pode conduzir a árvores menos balanceadas. A restrição (44) é forte e pode conduzir rapidamente a uma solução inteira, mesmo não sendo a solução óptima. Já a restrição (45) poderá ter um impacto menor na relaxação linear. A regra

de partição que propomos nesta Secção tenta melhorar o balanceamento da árvore usando a correlação que existe entre as procuras dos itens e as multiplicidades dos padrões.

Em vez de restringir o valor de uma multiplicidade específica como na regra anterior, a regra de partição que propomos nesta Secção procura definir o número de multiplicidades que devem existir num determinado intervalo de valores. Esses intervalos não são escolhidos de qualquer forma, mas irão corresponder precisamente a intervalos entre valores de procuras sucessivas. Vamos designar por *B'* o vector de procuras construído com base nas procuras de cada um dos itens, e de tal forma que não haja nenhum valor repetido. Seja b_i' o $i^{ésima}$ valor de *B'*. Assumimos que os valores de *B'* estão ordenados por ordem crescente. Para cada intervalo definido pelos valores $b'_i + 1$ e b'_{i+1} (i=1,…, |*B'*|-1) e desde que ambos os valores sejam inferiores ou iguais a n^{max}, vamos calcular o valor X_i' da seguinte forma:

$$X'_i = \sum_{p \in P} \sum_{n=b'_i+1}^{b'_{i+1}} a_{np}^{mult} \delta_p \, .$$

Para i=0, temos que:

$$X'_0 = \sum_{p \in P} \sum_{n=1}^{b'_1} a_{np}^{mult} \delta_p .$$

O valor das variáveis $\delta_p, p \in P$, corresponde ao valor obtido na relaxação linear. Finalmente, se algum dos X'_i for fraccionário, iremos escolher aquele com o maior índice i. As restrições de partição a partir das quais são gerados os novos nodos de pesquisa são as seguintes:

$$\sum_{p \in P} \sum_{n=b'_i}^{b'_{i+1}} a_{np}^{mult} \delta_p \geq \lceil X'_i \rceil$$

e

$$\sum_{p \in P} \sum_{n=b'_i}^{b'_{i+1}} a_{np}^{mult} \delta_p \leq \lfloor X'_i \rfloor .$$

Como todas as regras que propomos neste Capítulo, esta regra de partição é robusta no sentido em que não dificulta o subproblema de geração de colunas associado ao primeiro bloco do modelo (24)-(29). Esse subproblema continua a poder ser resolvido através de um algoritmo de Programação Dinâmica em que os estados do modelo representam o número de rolos já usados e também o número de multiplicidades já escolhidas. Por outro lado, e como acontece com todas as restrições de partição que se irão basear em variáveis do primeiro bloco, esta regra também não garante a convergência para a solução óptima inteira, tendo de ser combinada com pelo menos uma outra regra de partição.

Exemplo 4.5

Na instância kT03, se ordenarmos as procuras por ordem crescente, obtemos a sequência de valores seguintes:

$$B' = \{\ 6, 10, 12, 18, 31, 48\ \}$$

A procura de valor igual a 6 (i=1) corresponde exclusivamente ao item de tamanho 50. O valor de X_0' é calculado com base nos valores de todas as combinações de multiplicidades com valor positivo porque todas elas usam multiplicidades com valor inferior a 6. O valor de X_0' é calculado da seguinte forma:

$$X_0' = 0{,}4 + 0{,}4 + 0{,}2 + \ (0{,}1 + 0{,}1) = \ 1{,}2.$$

A combinação de multiplicidades associada à coluna 17 usa duas multiplicidades com valor inferior ou igual a 6 (neste caso, igual a 6).

As restrições de partição que seriam adicionadas neste caso seriam:

$$\sum_{p \in P} \sum_{n=1}^{6} a_{np}^{mult} \delta_p \geq 2 \text{ e}$$

$$\sum_{p \in P} \sum_{n=1}^{6} a_{np}^{mult} \delta_p \leq 1\ .$$

■

4.2.6 Regra de Partição Baseada nas Variáveis do 1º Bloco: Esquema VI

A última regra de partição que investigámos nesta dissertação é próxima daquela que introduzímos na Secção anterior. O objectivo dessa nova regra é de aumentar o balanceamento da árvore de partição baseando as restrições de partição em conjuntos de variáveis de maior dimensão.

Vamos designar por X_i'' a variável que representa o número de multiplicidades usadas numa solução da relaxação linear de (24)-(29) desde a multiplicidade inicial 1 até à multiplicidade b_i' ($b_i' \leq n^{max}$). Temos assim que:

$$X_i'' = \sum_{p \in P} \sum_{n=1}^{b_i'} a_{np}^{mult} \delta_p.$$

Entre todos os X_i'' fraccionários, escolhemos aquele com o maior valor do índice i. As restrições de partição a partir das quais irão ser gerados os nodos da árvore são as seguintes:

$$\sum_{p \in P} \sum_{n=1}^{b_i'} a_{np}^{mult} \delta_p \geq \lceil X_i'' \rceil$$

e

$$\sum_{p \in P} \sum_{n=1}^{b_i'} a_{np}^{mult} \delta_p \leq \lfloor X_i'' \rfloor.$$

Exemplo 4.6

No caso da instância kT03 e da solução da relaxação linear representada na Figura 4.1, o valor de X_2'', por exemplo, indica o número de multiplicidades que são usadas até à multiplicidade de valor igual a 10 (a 2ª procura mais pequena do conjunto de itens). O valor de X_2'' é igual a 2,2. As restrições de partição associadas a esta variável seriam:

$$\sum_{p \in P} \sum_{n=1}^{10} a_{np}^{mult} \delta_p \geq 3$$

e

$$\sum_{p \in P} \sum_{n=1}^{10} a_{np}^{mult} \delta_p \leq 2.$$

■

4.3 Algoritmo de Partição e Geração de Colunas: Implementação

Os algoritmos de partição e geração de colunas que foram implementados no âmbito desta dissertação seguem os passos gerais que passamos a descrever. A relaxação linear do modelo (24)-(29) é inicializada apenas com uma coluna artificial de custo muito elevado e que garante que todas as restrições podem sempre ser satisfeitas qualquer que seja o nodo da árvore de partição. As colunas associadas às combinações de multiplicidades são geradas através de um algoritmo de Programação Dinâmica em que os estados identificam quer o número de rolos já usados, quer o número de multiplicidades já usada na combinação. Usámos um algoritmo semelhante àquele que foi usado em [AMVC09]. As colunas associadas aos padrões de corte são geradas usando um algoritmo semelhante ao que foi usado em [AVC08]. Para reforçar a relaxação linear, considerámos também o uso de cortes derivados a partir de funções duais válidas. Os esquemas de reforço descritos na Secção 3.6 foram também considerados.

Como referimos nas Secções anteriores, alguns esquemas não garantam por si só que uma solução inteira seja determinada no final da execução. Dado que o único esquema que garante a convergência para uma solução inteira é o esquema I, ele foi sempre usado em último recurso, quando não é possível derivar restrições de partição a partir de nenhum dos outros esquemas.

Adicionalmente, usámos uma heurística de arredondamento similar à que é descrita por Vanderbeck em [V00]. Com essa heurística, tentámos melhorar o valor da incumbente com

base na solução da relaxação linear do problema mestre. O nosso objectivo é o de comparar essas regras de partição com o algoritmo proposto por Vanderbeck em condições próximas.

O algoritmo que desenvolvemos percorre a árvore de partição seguindo uma estratégia de pesquisa em profundidade. Com essa abordagem, pretende-se melhorar rapidamente o valor das soluções incumbentes. Em cada nodo da árvore de partição, continuam também a serem gerados cortes a partir de funções duais válidas.

4.4 Conclusões

Neste Capítulo, apresentámos e descrevemos várias regras de partição para um algoritmo de partição e geração de colunas baseado no modelo (24)-(29). Esse modelo foi usado em [AMVC09] apenas para derivar limites inferiores. As experiências que apresentamos no próximo Capítulo são os primeiros resultados da resolução exacta desse modelo de Programação Inteira.

Todos os esquemas descritos neste Capítulo são robustos no sentido em que não dificultam a resolução dos subproblemas de geração de colunas. Em qualquer nodo da árvore de partição a complexidade dos subproblemas é igual à complexidade dos subproblemas na raiz da árvore.

Capítulo 5

Experiências Computacionais

5.1 Introdução

Neste Capítulo, descrevemos os resultados das experiências computacionais que realizámos para testar os diferentes esquemas de partição descritos no Capítulo anterior. O nosso objectivo foi de comparar entre eles esses vários esquemas. Usámos como referência as instâncias de Vanderbeck [V00]. Essas instâncias são instâncias reais de pequena e média dimensão que têm sido usadas na literatura para comparar métodos de resolução para o PMP.

A Tabela 5.1 descreve as principais características dessas instâncias. A coluna *Instância* identifica o nome de cada uma das instâncias, *m* representa o número de itens diferentes, *LB* e *UB* designam respectivamente o melhor limite inferior e superior conhecido para a instância (quando LB=UB, a solução é óptima; apenas as 3 últimas instâncias não foram resolvidas até agora), UB_0 representa um limite superior inicial (obtido por exemplo resolvendo um problema de corte standard a uma dimensão e contando o número de padrões diferentes na respectiva solução) e z_{CSP} representa o menor número de rolos necessários para cortar os itens.

A Tabela 5.2 ilustra os resultados que foram obtidos por Vanderbeck [V00]. Não indicamos o valor dos tempos de computação obtidos por Vanderbeck. Esses resultados foram obtidos num computador com características significativamente inferiores ao do computador que usámos. O significado dos dados listados nessa tabela é o seguinte:

- LB : valor do melhor limite inferior obtido;

- UB: valor da solução incumbente (quando LB=UB, a solução encontrada é óptima);
- Nodos: número de nodos da árvore de partição que foram explorados;
- Max. Prof.: profundidade máxima que foi atingida na árvore de partição.

	Instância	***m***	***LB***	***UB***	UB_0	z_{CSP}
1	kT03	7	6	6	6	66
2	kT05	10	9	9	11	47
3	kT01	5	3	3	6	14
4	kT02	24	18	18	20	66
5	kT04	16	9	9	16	38
6	d16p6	16	9	9	14	38
7	7p18	7	6	6	8	91
8	d33p20	23	8	8	17	29
9	12p19	12	5	5	15	23
10	d43p21	32	10	10	26	40
11	kT06	9	4	4	15	51
12	kT07	11	5	5	16	65
13	14p12	14	5	5	18	56
14	kT09	14	5	6	24	110
15	11p4	11	4	5	24	101
16	30p0	26	7	8	28	90

Tabela 5.1: Dados das instâncias de Vanderbeck [V00]

	Instância	***LB***	***UB***	***nodos***	***Max. Prof.***
1	kT03	6	6	91	17
2	kT05	9	9	37	15
3	kT01	3	3	1	0
4	kT02	18	18	1	0
5	kT04	9	9	57	23
6	d16p6	9	9	39	18
7	7p18	6	7	1354	48
8	d33p20	8	8	655	36
9	12p19	5	5	47	14
10	d43p21	10	10	33	10
11	kT06	4	4	51	11
12	kT07	5	5	163	17
13	14p12	5	5	1	0
14	kT09	5	6	140	18
15	11p4	4	5	19	9
16	30p0	7	8	13	8
	Média	**7,1**	**7,3**	**170**	**15**

Tabela 5.2: Resultados do algoritmo de Vanderbeck [V00]

Conforme se pode observar na Tabela 5.2, o algoritmo de Vanderbeck não resolveu 4 das 16 instâncias. Essas instâncias são assinaladas a cinzento na Tabela 5.2. Para essas instâncias, a diferença entre os limites inferiores e superiores nunca ultrapassa 1. De referir que Vanderbeck para a execução do algoritmo se depois de 2 horas não for encontrada a solução óptima.

5.2 Resultados

Os testes computacionais foram realizados num computador com um processador Intel Core Duo de 2,20 GHz e 2GB de RAM. A execução do algoritmo de partção e geração de colunas é abortada se depois de 180 segundos na árvore de partição não tiver sido encontrada a solução óptima. Esse tempo foi escolhido tendo em conta o número elevado de execuções do algoritmo que foram realizadas.

De seguida, apresentamos os resultados que foram obtidos para a relaxação linear do modelo (24)-(29) e os resultados da aplicação dos diferentes esquemas de partição descritos no Capítulo anterior.

Resolução da relaxação linear

A relaxação linear do modelo (24)-(29) é resolvida como descrito em [AMVC09]. Adicionalmente, usámos as estratégias de reforço que foram descritas no Capítulo 3, e adicionámos cortes baseados em funções duais válidas [AVC08]. Os resultados são descritos na Tabela 5.3.

Na Tabela 5.3, z_{RL} representa o valor da solução da relaxação linear e t_{RL} o tempo necessário para resolver essa relaxação linear. O modelo (24)-(29) reforçado com todos os cortes que descrevemos é forte. Em 4 casos, o limite inferior corresponde ao valor da solução óptima do problema. Os tempos de computação são razoáveis para a maior parte das instâncias (abaixo de 30 segundos). As 3 instâncias finais são as mais difíceis do conjunto. Ainda não foi possível até hoje determinar a solução óptima dessas instâncias.

	Instância	z_{RL}	t_{RL}
1	kT03	5,01	0,195
2	kT05	8	1,287
3	kT01	3	0,055
4	kT02	18	0,691
5	kT04	8	1,668
6	d16p6	8	1,741
7	7p18	5	3,734
8	d33p20	7	2,581
9	12p19	4	3,488
10	d43p21	9	32,279
11	kT06	3	7,264
12	kT07	4	23,968
13	14p12	5	9,009
14	kT09	4	122,628
15	11p4	4	63,468
16	30p0	6	101,488

Tabela 5.3: Dados da relaxação linear para as instâncias de Vanderbeck [V00]

Esquema I de partição

Os resultados do esquema I de partição são descritos na Tabela 5.4. Nessa tabela, e nas seguintes, t_{BB} representa o tempo de computação em nodos da árvore de partição (excluindo a relaxação linear inicial).

O esquema I de partição permite resolver 7 das 16 instâncias do conjunto no tempo máximo que fixámos. Para as outras instâncias, o intervalo de optimalidade permanece elevado. Lembramos que não foram usadas heurísticas para determinar soluções inteiras a partir das soluções da relaxação linear nos nodos da árvore de partição. O nosso objectivo com estas experiências foi apenas o de avaliar qual dos esquemas de partição seria o mais adequado à resolução do modelo (24)-(29).

A profundidade média do nodo mais fundo da árvore (o que tem mais restrições de partição) é de 18,5. Esse resultado poderia indicar que seria necessário um número substancial de restrições de partição do 1º tipo para poder concluir que o ramo não é interessante. No

entanto, a pesquisa num nodo só pára quando o valor da solução nesse nodo é pior que o limite superior (incumbente). Como os limites superiores iniciais que foram usados são elevados (como na instância kT06, por exemplo), não é possível afirmar que as restrições de partição são fracas.

	Instância	***LB***	***UB***	***nodos***	***Max. Prof.***	t_{BB}
1	kT03	6	6	0	0	0
2	kT05	9	9	732	30	152,388
3	kT01	3	3	59	11	1,326
4	kT02	18	20	2619	35	181,043
5	kT04	9	9	1117	29	149,221
6	d16p6	8	10	1465	32	180,104
7	7p18	5	6	359	21	181,487
8	d33p20	8	8	266	28	116,099
9	12p19	5	5	169	26	144,939
10	d43p21	9	26	26	28	180,452
11	kT06	3	15	9	11	181,587
12	kT07	4	6	47	19	190,355
13	14p12	5	6	39	22	181,843
14	kT09	4	24	1	1	200,175
15	11p4	4	28	2	2	194,157
16	30p0	6	28	1	1	361,593
	Média	**6,63**	**13,06**	**431,94**	**18,5**	**162,30**

Tabela 5.4: Resultados para o Esquema I de partição

Esquema II

Os resultados do esquema II são indicados na Tabela 5.5. Esse esquema não parece trazer nenhuma melhoria. Com este esquema, apenas 2 das instâncias são resolvidas no tempo limite de 180 segundos na árvore de partição. Os limites inferiores não são melhorados, e permanecem iguais aos que são obtidos ao resolver a relaxação do problema original. Também ao nível da profundidade máxima que é atingida, os resultados não melhoram ao contrário do que se poderia esperar.

	Instância	*LB*	*UB*	*nodos*	*Max. Prof.*	t_{BB}
1	kT03	6	6	0	0	0
2	kT05	8	11	786	40	180,49
3	kT01	3	3	574	43	102,913
4	kT02	18	20	2751	39	180,024
5	kT04	8	16	1076	50	180,32
6	d16p6	8	14	1286	54	180,012
7	7p18	5	8	87	39	182,677
8	d33p20	7	17	246	74	180,187
9	12p19	4	15	134	64	181,758
10	d43p21	9	26	31	15	181,71
11	kT06	3	15	19	12	180,941
12	kT07	4	16	57	54	181,551
13	14p12	5	18	61	59	188,179
14	kT09	4	24	1	0	269,51
15	11p4	4	28	5	4	318,117
16	30p0	6	28	1	0	282,888
	Média	**6,38**	**16,56**	**444,69**	**34,19**	**185,70**

Tabela 5.5: Resultados para o Esquema II de partição

Esquema III

A Tabela 5.6 ilustra os resultados obtidos com o esquema de partição III. Os resultados são muito semelhantes aos resultados que foram obtidos com o esquema II. Apenas 2 instâncias são resolvidas. O melhor esquema de partição baseado em variáveis de fluxos em arcos é, para estas instâncias, o Esquema I.

Esquema IV

O esquema IV é o primeiro esquema de partição que não se baseia exclusivamente em variáveis de fluxo. Lembramos que este esquema não é suficiente para garantir que uma solução óptima inteira é encontrada no final. Por essa razão, o esquema é combinado com o esquema I. O esquema I é usado quando não foi possível identificar uma solução óptima inteira, e não é possível derivar restrições de partição a partir das regras do esquema IV.

Os resultados obtidos com esse esquema são indicados na Tabela 5.7.

	Instância	*LB*	*UB*	*nodos*	*Max. Prof.*	t_{BB}
1	kT03	6	6	0	0	0
2	kT05	8	11	1029	44	180,06
3	kT01	3	3	341	44	4,029
4	kT02	18	20	2291	28	183,301
5	kT04	8	16	1740	50	180,005
6	d16p6	8	14	1723	49	180,042
7	7p18	5	7	545	66	180,077
8	d33p20	7	10	804	42	180,765
9	12p19	4	6	532	50	180,063
10	d43p21	9	26	40	25	181,029
11	kT06	3	15	32	20	182,115
12	kT07	4	16	122	30	180,272
13	14p12	5	6	77	37	180,13
14	kT09	4	24	1	0	217,275
15	11p4	4	28	4	1	190,398
16	30p0	6	28	1	0	314,901
	Média	**6,38**	**14,75**	**580,13**	**30,38**	**6,38**

Tabela 5.6: Resultados para o Esquema III de partição

	Instância	*LB*	*UB*	*nodos*	*Max. Prof.*	t_{BB}
1	kT03	6	6	0	0	0
2	kT05	9	9	145	24	38,058
3	kT01	3	3	12	5	0,278
4	kT02	18	20	2661	31	180,117
5	kT04	9	9	92	20	15,791
6	d16p6	9	9	102	21	15,11
7	7p18	5	8	442	23	180,61
8	d33p20	8	8	60	28	36,903
9	12p19	5	5	89	26	63,509
10	d43p21	10	26	28	19	180,814
11	kT06	3	15	12	5	194,242
12	kT07	4	16	21	8	182,131
13	14p12	5	18	64	21	181,006
14	kT09	4	24	2	1	278,801
15	11p4	4	28	10	5	183,903
16	30p0	6	28	2	1	411,951
	Média	**6,75**	**14,50**	**233,88**	**14,88**	**133,95**

Tabela 5.7: Resultados para o Esquema IV de partição

Como se pode observar, o número de instâncias que são resolvidas é igual ao número de instâncias resolvidas com o esquema I. Contudo, ao combinar os dois esquemas, foi possível reduzir substancialmente o número de nodos da árvore de partição necessários para encontrar a solução óptima inteira (de 431,94, passamos para 233,88). Os tempos de computação também foram reduzidos em consequência da diminuição do número de nodos.

Esquema V

Os resultados obtidos com o esquema de partição V são indicados na Tabela 5.8. Os resultados não melhoram comparativamente com o esquema anterior.

	Instância	***LB***	***UB***	***nodos***	***Max. Prof.***	t_{BB}
1	kT03	6	6	0	0	0
2	kT05	9	9	67	13	18,926
3	kT01	3	3	26	24	0,812
4	kT02	18	20	2447	32	181,016
5	kT04	9	9	85	20	18,209
6	d16p6	9	9	614	26	81,247
7	7p18	5	7	481	52	180,64
8	d33p20	8	8	117	34	70,553
9	12p19	5	5	226	26	158,451
10	d43p21	9	26	26	13	187,808
11	kT06	3	15	15	6	182,948
12	kT07	4	16	17	7	184,869
13	14p12	5	18	55	21	180,099
14	kT09	4	24	1	0	190,128
15	11p4	4	28	4	2	244,136
16	30p0	6	28	1	0	365,434
	Média	**6,69**	**14,44**	**261,38**	**17,25**	**140,33**

Tabela 5.8: Resultados para o Esquema V de partição

Esquema VI

Finalmente, os resultados do esquema VI são ilustrados na Tabela 5.9. Esse esquema permite resolver 8 das 16 instâncias do conjunto no tempo limite. O número médio de nodos necessários diminui também de forma substancial.

	Instância	*LB*	*UB*	*nodos*	*Max. Prof.*	t_{BB}
1	kT03	6	6	0	0	0
2	kT05	9	9	59	22	14,907
3	kT01	3	3	12	5	0,281
4	kT02	18	18	23	13	0,946
5	kT04	9	9	85	19	13,519
6	d16p6	9	9	12	8	4,377
7	7p18	5	8	462	23	180,358
8	d33p20	8	8	69	45	38,24
9	12p19	5	5	86	33	75,033
10	d43p21	10	26	22	17	184,499
11	kT06	3	15	10	5	205,215
12	kT07	4	16	16	8	191,243
13	14p12	5	18	57	18	182,029
14	kT09	4	24	2	1	348,146
15	11p4	4	28	7	5	190,327
16	30p0	6	28	2	1	428,86
	Média	**6,75**	**14,38**	**57,75**	**13,94**	**128,62**

Tabela 5.9: Resultados para o Esquema VI de partição

Os resultados do esquema VI são claramente os melhores resultados obtidos entre os 6 esquemas de partição que foram testados.

5.3 Conclusões

Neste Capítulo, apresentámos os resultados que foram obtidos com os esquemas de partição discutidos no Capítulo anterior. Usámos instâncias reais do Problema de Minimização de Padrões que foram usadas por vários autores na literatura. Os melhores resultados foram obtidos com o esquema de partição VI. Esse esquema combina restrições de partição baseadas em variáveis de fluxos em arcos, e em variáveis associadas às combinações de multiplicidades.

O nosso objectivo não foi de comparar os nossos resultados com os de Vanderbeck. Por essa razão, optámos por usar um tempo limite de computação relativamente reduzido (180

segundos na árvore de partição). Essa escolha ficou-se a dever a vários factores. Em primeiro lugar, Vanderbeck usa heurísticas de arredondamento para tentar melhorar o valor da solução incumbente. Essa abordagem não permite ter uma ideia clara relativamente à eficiência do esquema de partição que é usado. No nosso algoritmo, não usámos essas heurísticas. Por outro lado, a diferença entre os computadores usados torna impossível a comparação entre os tempos de execução dos algoritmos.

Capítulo 6

Conclusões

O Problema de Minimização de Padrões é um problema relevante na prática e cuja resolução pode conduzir a economias significativas. O problema é um problema de corte no qual se consideram custos de *setup*. Trata-se de um problema de grande complexidade que tem sido resolvido principalmente usando heurísticas. Os métodos de resolução exacta que foram propostos até agora permitem apenas resolver instâncias de pequena e média dimensão.

Nesta dissertação, discutímos os diferentes modelos de Programação Inteira que foram propostos na literatura, nomeadamente o de Vanderbeck [V00], o modelo baseado numa extensão do modelo de Gilmore e Gomory [GG61] e o modelo de fluxos de Alves e Carvalho [AVC08]. Descrevemos os algoritmos associados a estes modelos assim como os métodos que foram usados para reforçar os modelos.

Explorámos um novo modelo de geração de colunas que foi proposto por Alves *et al.* em [AMVC09]. Descrevemos formas de reforçar esse modelo. Uma das estratégias que propomos passa pelo uso de funções duais válidas. Essa abordagem já tinha sido usada em [AVC08]. Nesta dissertação, mostrámos como é que podiam ser obtidos cortes de melhor qualidade combinando restrições que são válidas para o problema. Apresentámos ainda uma nova família de cortes para o modelo do artigo [AMVC09].

Estudámos também estratégias para determinar soluções óptimas inteiras. Discutímos vários esquemas de partição. Os esquemas foram testados através de experiências computacionais em instâncias da literatura. O esquema com o qual foram obtidos os melhores

resultados baseia-se em restrições de partição que são derivadas a partir das variáveis de decisão dos dois blocos do modelo de geração de colunas descrito em [AMVC09].

O nosso estudo teve por objectivo determinar o melhor esquema de partição entre os vários que foram propostos. Por esse motivo, não usámos heurísticas ou qualquer outro mecanismo que pudesse acelerar o processo de resolução.

Com esta dissertação, contribuímos com resultados relativamente à resolução exacta do Problema de Minimização de Padrões usando o modelo descrito em [AMVC09]. A obtenção de soluções óptimas inteiras com base nesse modelo nunca tinha sido considerada antes. Apesar dos resultados alcançados, muitas instâncias entre aquelas que considerámos nas nossas experiências permaneceram não resolvidos, mesmo quando o tempo de resolução é alargado.

Os algoritmos poderiam ser melhorados se fossem usadas estratégias adicionais de aceleração. O uso de heurísticas de arredondamento ao nível dos nodos da árvore de pesquisa, à semelhança do que foi feito por Vanderbeck, poderia trazer benefícios. A exploração desses mecanismos é deixada como trabalho futuro.

Referências

[AMVC09] C. Alves, R. Macedo, J. Valério de Carvalho, “New Lower Bounds Based on Column Generation and Constraint Programming for the Pattern Minimization Problem”, Computers and Operations Research (in press), 2009.

[AVC08] C. Alves, J. Valério de Carvalho, “A branch-and-price-and-cut algorithm for the pattern minimization problem”, *RAIRO Operations Research*, 42, pp. 435-453, 2008.

[B03] G. Belov. Problems, “Models and Algorithms in One- and Two- Dimensional Cutting”, Tese de Doutoramento, Universidade de Dresden, 2003.

[D90] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research,* vol. 44, pp. 145-159, 1990.

[D93] A. Diegel, M. Chetty, S. Van Schalkwyck, S. Naidoo, “Setup combining in the trim loss problem - 3-to-2 & 2-to-1”. Working paper, Business Administration, University of Natal, Durban.

[FS01] S. Fekete, J. Schepers, “New classes of lower bounds for bin-packing problems”, *Mathematical Programming*, vol. 91, pp. 11-31, 2001.

[FW00] H. Foerster, G. Wäscher, “Pattern reduction in one-dimensional cutting stock problem”, *International Journal of Production Research*, vol. 38, pp.1657–1676, 2000.

[GG61] P. C. Gilmore, R. E. Gomory, "A Linear Programming Approach to the Cutting-Stock Problem " *Operations Research,* vol. 9, pp. 849-859, 1961.

[GG63] P. C. Gilmore, R. E. Gomory, "A Linear Programming Approach to the Cutting Stock Problem - Part II," *Operations Research,* vol. 11, pp. 863-888, 1963.

[GG65] P. C. Gilmore, R. E. Gomory, "Multistage Cutting Stock Problems of Two and More Dimensions," *Operations Research,* vol. 13, no.1, pp. 94-120, 1965.

[H75] R. Haessler, “Controlling cutting pattern changes in one-dimensional trim problems”, *Operations Research*, vol. 23, pp. 483–93, 1975.

[K60] L. V. Kantorovich, “Mathematical models of organising and planning production”, *Management Science*, 6:366–422, 1960. Traduzido a partir do original em russo, 1939.

[NW99] G. Nemhauser, L. Wolsey, “Integer and Combinatorial Optimization”, Wiley-Interscience, 1999.

[V00] F. Vanderbeck, “Exact algorithm for minimising the number of setups in the onedimensional cutting stock problem”, *Operations Research*, 48(6):915–926, 2000.

[Y06] H. Yanasse, M. Limeira, “A hybrid heuristic to reduce the number of different patterns in cutting stock problems”, *Computers and Operations Research*, vol.33, pp. 2744–2756, 2006.

[WHS07] G. Wäscher, H. Haußner, H. Schumann, "An Improved Typology of Cutting and Packing Problems," *European Journal of Operational Research,* 183(3):1109-1130, 2007.