2
Técnicas de Resolução

Um problema de otimização combinatória é definido por um par $(E, \mathcal{F})$ e por um vetor de custos $c \in \mathbb{R}^E$, onde E é o conjunto base e $\mathcal{F} \subseteq 2^E$, uma família de subconjuntos de E, é o conjunto de soluções viáveis. Cada subconjunto $S \in \mathcal{F}$ é associado a um custo $c(S) = \sum_{e \in S} c_e$. O objetivo é encontrar a solução $S \in \mathcal{F}$ de menor (ou maior) custo.

A resolução deste tipo de problemas, entendida como a obtenção de uma solução garantidamente ótima (em contraste com apenas obter uma solução de qualidade aceitável usando técnicas heurísticas), pode ser bastante difícil. Embora geralmente seja conceitualmente fácil enumerar os elementos de $\mathcal{F}$, esse conjunto costuma ser extremamente grande. Isso impede que a simples enumeração seja uma abordagem viável de solução. As abordagens que vêm apresentando maior sucesso são as que utilizam técnicas de programação inteira. Para tal, cada elemento do conjunto $\mathcal{F}$ é associado a uma posição de um vetor. Assim, uma solução viável é descrita como um vetor de incidência que satisfaz um conjunto de restrições.

Um problema geral de Programação Inteira pode ser definido como:

$$min f(x),\ x \in \mathcal{F} \subseteq \mathbb{Z}^n_+, \tag{2-1}$$

onde f é chamada de função objetivo e $\mathcal{F}$ é o conjunto de soluções associadas ao problema. Sendo linear a função objetivo $f(x)$ do problema e $\mathcal{F}$ definido por um sistema de restrições lineares sobre variáveis inteiras, tem-se um problema de programação linear inteira (*IP - Linear Integer Programming Problem*):

$$\min\{f(x) = c.x \mid A.x \geq b,\ x \in \mathbb{Z}^n_+\}, \tag{2-2}$$

onde c, b e A são matrizes de dimensões $1 \times n$, $m \times 1$ e $m \times n$, respectivamente. O conjunto de soluções viáveis do *IP* é definido por:

$$\mathcal{F} = \{x \in \mathbb{Z}^n_+ \mid A.x \geq b\}\ . \tag{2-3}$$

O problema de programação linear (*LP - Linear Programming Problem*) é definido como:

$$\min\{f(x) = c.x \mid A.x \geq b,\ x \in \mathbb{R}^n_+\} \tag{2-4}$$

e o seu conjunto de soluções é dado por:

$$\mathcal{F} = \{x \in \mathbb{R}^n_+ \mid A.x \geq b\} \tag{2-5}$$

A programação linear inteira é um modelo muito abrangente e, embora existam técnicas gerais que lidem com essa estrutura, muito trabalho tem sido desenvolvido no intuito de obter-se algoritmos especiais para resolver subclasses particulares desse modelo geral. Muitos desses algoritmos baseiam-se em uma metodologia enumerativa (geralmente uma variação do algoritmo *branch-and-bound* descrito na seção 2.2.1) e utilizam uma relaxação do problema 2-2 para obter em tempo razoável uma estimativa otimista (um limite inferior) para o valor da melhor solução que pode ser encontrada em cada ramo da enumeração. Esta estimativa é usualmente obtida a partir da relaxação linear (ou outras relaxações) do *IP* em questão.

A relação entre o tempo para o cálculo dessas estimativas e a sua qualidade, que pode ser medida pela proximidade ao valor da solução ótima, determinam a eficiência do método enumerativo. Deste modo, a pesquisa na resolução de problemas combinatórios concentra-se na busca de formulações que forneçam as melhores estimativas possíveis. Duas linhas de pesquisas têm sido exploradas com este objetivo.

Uma dessas linhas de pesquisas procura, a partir da formulação de um problema como um *IP*, caracterizar classes de desigualdades adicionais para que o conjunto de soluções de sua relaxação linear se aproxime mais do envoltório convexo $conv(\mathcal{F})$ das soluções deste *IP*.

De um modo geral, estas classes possuem um número exponencial de desigualdades, o que não quer dizer que sua utilização seja proibitiva. De fato, é neste contexto que o termo "método de planos de corte" se insere. Nele as desigualdades são incluídas na formulação do programa linear na medida em que se tornam necessárias (violadas). Para determinar quando isso acontece, resolve-se um problema dito "de separação". Isto é, um problema onde, dada a solução de um programa linear que contém apenas parte das desigualdades, determina-se se existe alguma desigualdade da classe que está violada. O método prossegue após a inclusão desta desigualdade no programa linear corrente.

Existem casos de classes de desigualdades em que o problema de

separação possui algoritmo de complexidade polinomial para sua resolução e outros nos quais a separação é um problema $\mathcal{NP}$-difícil. Neste segundo caso, usualmente faz-se uma separação heurística, pois o objetivo de sua utilização é apenas melhorar a estimativa correspondente.

Uma segunda forma de se obter estimativas melhores advém de métodos de relaxação Lagrangeana ou da, de certa forma equivalente, decomposição de *Dantzig-Wolfe* (veja seção 2.1.1). Neste caso, os *LP*s correspondentes possuem usualmente um número exponencial de variáveis.

A resolução de tais *LP*s é realizada com um método usualmente chamado de "geração de colunas" (seção 2.1.2). Este método consiste tão somente no conhecido método *Simplex* revisado, com a diferença de que, a cada iteração, as colunas candidatas a entrar na base são obtidas através da resolução de um subproblema auxiliar, ao invés da usual inspeção. Ao contrário do problema de separação, o problema de geração de colunas precisa ser resolvido até a otimalidade, uma vez que as estimativas assim obtidas somente são válidas quando a solução ótima do programa linear é encontrada. Na literatura, esses problemas possuem freqüentemente algoritmos de complexidade pseudo-polinomial para sua resolução, e este será o caso ao longo desta dissertação.

A seção 2.1 descreve a decomposição de problemas de programação linear inteira (*IPs*) para gerar formulações que permitam considerar apenas implicitamente o conjunto total de variáveis do problema. Na seção 2.2, que trata da resolução de *IPs* por algoritmos enumerativos, são descritos o algoritmo *branch-and-bound* e algumas de suas variações, que utilizam uma, ou ambas, dessas formas de se melhorar a estimativa do valor da melhor solução em cada ramo da enumeração.

2.1 Reformulação de *IPs*

O objetivo desta primeira parte do texto é apenas introduzir os conceitos básicos de programação inteira e de algumas das principais metodologias utilizadas na resolução de problemas de programação inteira. Espera-se que o leitor desta dissertação tenha conhecimento básico dos fundamentos da programação linear e do algoritmo *Simplex* para resolução de *LP*s. Maiores detalhes a respeito desses tópicos podem ser obtidos, por exemplo, em [23, 60, 70, 62, 19, 24, 81] ou nas demais referências citadas ao longo deste capítulo.

O desenvolvimento de métodos de resolução de problemas lineares que explorem a estrutura particular de determinado problema foi sugerido inicialmente por Ford e Fulkerson [34]. Este trabalho é considerado por muitos o marco inicial da abordagem de geração de colunas e reconhecidamente inspirou Dantzig e Wolfe [28] a proporem um esquema de decomposição de problemas lineares que generaliza essa abordagem (veja seção 2.1.1).

Essa decomposição basicamente consiste na escrita de grupos de variáveis como uma combinação convexa de pontos extremos. O grande mérito desse esquema é retirar parte das restrições do problema linear, em geral restrições com estruturas bem definidas, e tratá-las a parte em um problema auxiliar. Dessa forma pode haver uma grande redução no número de restrições do problema resultante, que passa, porém, a conter um número potencialmente exponencial de variáveis.

O método de decomposição de *Dantzig-Wolfe* foi uma importante ferramenta para a resolução de modelos de programação linear de grande porte, que não podiam ser resolvidos por *softwares* que implementavam o método *Simplex* padrão por excederem algum limite e/ou capacidade dos mesmos. Com o avanço na implementação desses *softwares* e o enorme progresso no desenvolvimento de novos *hardwares* (tanto em termos de velocidade de CPU, quanto na disponibilidade de memória), este método tornou-se menos popular.

O trabalho pioneiro na formulação de um problema de programação inteira com um grande número de variáveis e sua resolução com técnicas de geração implícita de colunas foi apresentado por Gilmore e Gomory [40, 41].

A associação da técnica de geração de colunas com o algoritmo *branch-and-bound* (veja seção 2.2.3), proposta por Desrosiers, Soumis e Desrochers [30], reforça a importância da decomposição de *Dantzig-Wolfe* e desde então a abordagem por geração de colunas torna-se uma das principais técnicas para resolução de problemas de programação inteira. Assim, na próxima seção são descritas as técnicas básicas de decomposição de problemas de programação linear e inteira e de geração de colunas

2.1.1 Decomposição de *Dantzig-Wolfe* para *LP*s

O método de decomposição de *Dantzig-Wolfe* [28], para problemas de programação linear, procura aproveitar a estrutura especial da matriz de restrições de um problema, criando um problema equivalente com menos restrições, entretanto, com grande número de variáveis.

Considere o problema de programação linear com n variáveis a seguir:

$$LP = \begin{cases} Z_{LP} = \min c.x \\ \text{sujeito a} \\ \quad A.x = b \qquad (1) \\ \quad D.x \leq d \qquad (2) \\ \quad x \geq 0\,. \end{cases}$$

O princípio do método de decomposição de *Dantzig-Wolfe* permite representar as soluções de um certo domínio convexo limitado como combinação linear convexa de seus pontos extremos. Assim, uma vez identificado esse domínio, as restrições que o definem são substituídas por um outro conjunto de restrições que definem a combinação linear de seus pontos extremos.

Suponha que $\mathcal{Q} = \{x_1, x_2, \ldots, x_p\}$ é um conjunto finito com os p pontos extremos do conjunto $\{x \in \mathbb{R}^n_+ \mid Dx \leq d\}$. Considere Q uma matriz de dimensão $n \times p$, em que cada coluna corresponde a um elemento do conjunto $\mathcal{Q}$. Assim, cada elemento do conjunto $\{x \in \mathbb{R}^n_+ \mid Dx \leq d\}$ pode ser representado por pelo menos um vetor λ através da seguinte equivalência:

$$\mathcal{X} = \begin{cases} x = Q.\lambda \\ \text{sujeito a} \\ \quad \mathtt{1}.\lambda = 1 \\ \quad \lambda \in \mathbb{R}^p_+\,. \end{cases}$$

A decomposição de *Dantzig-Wolfe* consiste na troca das variáveis x, no problema LP, por sua expressão equivalente dada por $\mathcal{X}$. A formulação resultante é habitualmente designada por Problema Mestre ou *Dantzig-Wolfe Master* (DWM) e, naturalmente, os valores Z_{LP} e Z_{DWM} são iguais:

$$DWM = \begin{cases} Z_{DWM} = \min (c.Q).\lambda \\ \text{sujeito a} \\ \quad (A.Q).\lambda = b \\ \quad \mathtt{1}.\lambda = 1 \\ \quad \lambda \geq 0\,. \end{cases}$$

O caso em que o domínio é ilimitado não será considerado nesta dissertação. Contudo, o teorema da representação de Minkowski e Weyl (veja [70] ou [62]) garante que a sua envoltória convexa é um poliedro representado por uma combinação convexa de um conjunto finito de pontos extremos e uma combinação linear de um conjunto finito de raios extremos. Uma de-

talhada descrição deste caso é fornecida, por exemplo, por Vanderbeck em [73, 75].

Caso bloco-diagonal

Se a matriz D tem uma estrutura bloco-diagonal, que permite reescrever as restrições $Ax = b$ e $Dx \leq d$ da seguinte forma:

$$\begin{array}{cccccc} A_1 & A_2 & \dots & A_k & x & = & b \end{array}$$
$$\begin{pmatrix} D_1 & & & & \\ & D_2 & & & \\ & & \ddots & & \\ & & & D_k & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} \leq \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{pmatrix} ,$$

então a formulação DWM pode ser detalhada para considerar essa estrutura especial da matriz D. Para tal, deve ser feita a definição de conjuntos $\mathcal{Q}_i$ apropriados para cada subconjunto de restrições $D_i x_i \leq d_i$ e respectivas relações $\mathcal{X}_i$, para $i = 1, \dots k$, e a posterior substituição de cada variável x_i pela sua forma equivalente $x_i = Q_i.\lambda_i$:

$$DWM' = \begin{cases} Z_{DWM'} = \min \sum_{i=1}^{k} (c_i.Q_i).\lambda_i \\ \text{sujeito a} \\ \quad \sum_{i=1}^{k} (A_i.Q_i).\lambda_i = b \\ \quad\quad 1.\lambda_i = 1 \quad i = 1, \dots, k \\ \quad\quad \lambda_i \geq 0 \quad i = 1, \dots, k \end{cases}$$

2.1.2 Geração de colunas

A geração de colunas é uma técnica para a resolução de problemas de programação linear com um grande número de variáveis, mais do que seria possível armazenar na memória de um computador. Esta técnica se faz necessária para resolver os problemas que resultam da decomposição de *Dantzig-Wolfe* (veja seção 2.1.1).

Considere o problema DWM definido na seção 2.1.1. A resolução direta desse problema em geral é impraticável, devido ao elevado número de variáveis λ nessa formulação. Entretanto, a grande maioria dessas variáveis assume valor zero em uma solução ótima e, portanto, não precisariam

fazer parte da formulação do problema (supondo que se saiba de antemão identificar essas variáveis). De fato, a teoria de programação linear garante que, caso não haja variáveis com limite superior, como em DWM, o número de variáveis não nulas em qualquer solução básica está limitado ao número de restrições. A técnica de geração de colunas usa este fato, reduzindo a solução do DWM à resolução de uma seqüência de programas lineares que contém apenas um pequeno subconjunto das variáveis do problema original.

Para qualquer matriz Q' que contenha um subconjunto das colunas da matriz Q, o seguinte problema linear é chamado de problema mestre restrito:

$$DWMR = \begin{cases} Z_{DWMR} = \min\,(c.Q').\lambda \\ \text{sujeito a} \\ \quad (A.Q').\lambda = b \qquad (1) \\ \quad\quad \mathtt{1}.\lambda = 1 \qquad (2) \\ \quad\quad\quad \lambda \geq 0\,. \end{cases}$$

Considere o $DWMR$ com um subconjunto Q' das colunas de Q suficiente para que $DWMR$ tenha solução viável (se necessário pode-se utilizar colunas artificiais não presentes em Q, com custo elevado, para obter essa viabilidade). Dada a solução ótima para tal problema, sejam μ e ν os vetores de multiplicadores duais ótimos associados às restrições (1) e (2), respectivamente. Variáveis com custo reduzido negativo com respeito a μ e ν são candidatas a serem adicionadas ao $DWMR$. A operação de encontrar tais variáveis é usualmente chamada de *pricing* e dá-se pela resolução do seguinte subproblema:

$$PS = \begin{cases} v(\mu,\nu) = \min\,(c - \mu.A).x - \nu \\ \text{sujeito a} \\ \quad D.x \leq d \\ \quad\quad x \in \mathbb{R}^n_+\,. \end{cases}$$

O algoritmo para encontrar a solução ótima do problema mestre DWM alterna entre a resolução da versão restrita $DWMR$ e a do subproblema gerador de colunas PS. Na resolução deste último, se $v(\mu,\nu) < 0$, então a solução ótima x^* define uma nova coluna ($q'_j = (A.x^*, 1)$ e $c_j = c.x^*$) que é adicionada a matriz Q' de $DWMR$. Esse processo é repetido até que $v(\mu,\nu) \geq 0$ (na verdade $v(\mu,\nu)$ sempre terminará com o valor zero, uma vez que as colunas da base ótima têm custo reduzido zero e são soluções válidas de PS). Neste ponto a solução corrente de $DWMR$ também é solução ótima de DWM (o valor das variáveis de DWM não presentes em $DWMR$ é 0).

Considere o caso em que a matriz D tem uma estrutura bloco-diagonal e a decomposição foi realizada de modo a aproveitar essa estrutura (problema DWM' definido na seção 2.1.1). Seja Q'_i uma matriz que contenha um subconjunto das colunas da matriz Q_i, para $i = 1, \ldots, k$ e $DWMR'$ o problema mestre restrito definido com tais matrizes:

$$DWMR' = \begin{cases} Z_{DWMR'} = \min \sum_{i=1}^{k} (c_i.Q'_i).\lambda_i \\ \text{sujeito a} \\ \quad \sum_{i=1}^{k} (A_i.Q'_i).\lambda_i = b & (1) \\ \quad\quad 1.\lambda_i = 1 \quad i = 1, \ldots, k & (2) \\ \quad\quad \lambda_i \geq 0 \quad i = 1, \ldots, k \end{cases}$$

A operação de *pricing* deve agora verificar, em separado, a existência de variáveis de custo reduzido negativo em cada bloco de colunas. Assim, se μ_i e ν_i são os vetores de multiplicadores duais associados às restrições (1) e (2) de $DWMR'$, respectivamente, essa operação deve resolver individualmente os k subproblemas definidos a seguir:

$$PS' = \begin{cases} v(\mu_i, \nu_i) = \min (c_i - \mu_i.A).x_i - \nu_i \\ \text{sujeito a} \\ \quad D_i.x_i \leq d_i \\ \quad\quad x_i \in \mathbb{R}^n_+ . \end{cases}$$

2.1.3 Reformulação de *IP*s

O método de decomposição de *Dantzig-Wolfe* para problemas de programação linear e o esquema de geração de colunas, mostrados nas seções 2.1.1 e 2.1.2, têm seus equivalentes para problemas de programação linear inteira. Assim como na decomposição de *Dantzig-Wolfe*, o princípio básico também é a representação de um conjunto convexo por uma combinação de seus pontos extremos.

Considere o seguinte problema de programação linear inteira com n variáveis:

$$IP = \begin{cases} Z_{IP} = \min c.x \\ \text{sujeito a} \\ \quad A.x = b \\ \quad D.x \leq d \\ \quad\quad x \in \mathbb{Z}^n_+ . \end{cases}$$

Suponha que o conjunto $\mathcal{Q} = \{x \in \mathbb{Z}_+^n \mid Dx \leq d\}$ é um conjunto finito com p elementos, ou seja, $\mathcal{Q} = \{x_1, x_2, ...x_p\}$. Observe que se $x \in \{0,1\}$, então $\mathcal{Q}$ coincide com os pontos extremos de sua envoltória convexa $conv(\mathcal{Q})$. Novamente, se $\mathcal{Q}$ não é limitado, é possível a sua representação em termos de seus pontos e raios extremos. Veja [73, 75] para uma detalhada descrição deste caso.

Considere Q uma matriz de dimensão $n \times p$, em que cada coluna corresponde a um elemento do conjunto $\mathcal{Q}$. Para cada x em $\mathcal{Q}$, existe um único vetor λ satisfazendo a seguinte relação:

$$\mathcal{X} = \begin{cases} x = Q.\lambda \\ \text{sujeito a} \\ \quad 1.\lambda = 1 \\ \quad \lambda \in \{0,1\}^p . \end{cases}$$

A reformulação de problemas de programação inteira, proposta por Gilmore e Gomory [40, 41], consiste na troca das variáveis x, no problema IP, por sua expressão equivalente dada por $\mathcal{X}$. O resultado é o novo problema inteiro IP' mostrado a seguir.

$$IP' = \begin{cases} Z_{IP} = \min (c.Q).\lambda \\ \text{sujeito a} \\ \quad (A.Q).\lambda = b \\ \quad 1.\lambda = 1 \\ \quad \lambda \in \{0,1\}^p . \end{cases}$$

O problema DWM (*Dantzig-Wolfe Master*), similar ao definido na seção 2.1.1, é obtido relaxando-se as restrições de integralidade em IP'. Resolver o problema DWM equivale a resolver o problema definido a seguir:

$$DWM = \begin{cases} Z_{DWM} = \min c.x \\ \text{sujeito a} \\ \quad A.x = b \\ \quad x \in Conv(D.x \leq d;\ x \in \mathbb{Z}_+^n) . \end{cases}$$

Seja Z_{LP} o valor obtido resolvendo-se a relaxação linear do problema IP. Sempre é verdade que $Z_{LP} \leq Z_{DWM} \leq Z_{IP}$ [39]. Entretanto, em muitas situações Z_{DWM} é bastante maior que Z_{LP}, fornecendo uma estimativa inferior muito mais forte para o valor de Z_{IP}. Essa é a principal motivação para o uso da decomposição de IPs.

A resolução do problema DWM, usando-se um esquema de geração de

colunas, é discutida na seção 2.1.2. A diferença básica é que agora a solução ótima x^*, do subproblema de geração de colunas, deve satisfazer restrições de integralidade ($x^* \in \mathbb{Z}_+^n$), ou seja, o subproblema PS é agora definido como:

$$PS = \begin{cases} v(\mu,\nu) = \min\,(c - \mu.A).x - \nu \\ \text{sujeito a} \\ \quad D.x \;\leq\; d \\ \quad\quad x \;\in\; \mathbb{Z}_+^n\,. \end{cases}$$

O problema de geração de colunas precisa ser resolvido até a otimalidade, uma vez que a solução ótima do problema $DWMR$ (veja seção 2.1.2) somente é uma estimativa válida para o valor da solução ótima do problema *IP'* se também for solução ótima de sua relaxação linear, ou seja, se também for solução ótima do problema DWM.

Essa abordagem somente é viável quando a estrutura do conjunto de restrições $D.x \leq d$ permite que o subproblema PS, que é um problema inteiro, seja resolvido em um tempo computacional razoável. Se o próprio subproblema PS é um problema da classe $\mathcal{NP}$-difícil e o mesmo não possui um algoritmo de complexidade ao menos pseudo-polinomial para sua resolução (o que não será o caso nas aplicações descritas na segunda parte desta tese), a abordagem não costuma ser viável.

Além disso, é essencial que no caso de adição de novos cortes ao problema DWM, o subproblema PS continue sempre com a mesma estrutura. Seja $\overline{\lambda}$ uma solução fracionária de DWM. Alguns pesquisadores, começando com Nemhauser e Park (1991) [63], tentaram melhorar o limite inferior dado por Z_{DWM} adicionando ao problema linear cortes $\pi^i.\lambda \leq b_i$ violados, ou seja, tais que $\pi^i.\overline{\lambda} > b_i$. Essa abordagem em geral não obtém bons resultados. Os cortes assim adicionados introduzem outros coeficientes e variáveis duais que devem ser levados em conta no novo problema de geração de colunas PS. Esses outros elementos destroem a estrutura de PS, tornando-o impossível de ser resolvido por algoritmos eficientes.

No final dos anos 90, alguns pesquisadores perceberam que cortes definidos sobre a variáveis originais x podiam ser adicionados a DWM sem alterar a estrutura do subproblema [77, 48, 49, 78, 32, 17]. Novamente, considere uma solução fracionária $\overline{\lambda}$ de DWM, que na formulação original IP corresponde à solução $\overline{x} = Q.\overline{\lambda}$. Um corte $a^i.x \leq b_i$ violado por $\overline{x}$ ($a^i.\overline{x} > b_i$) pode ser adicionado a DWM para cortar a solução $\overline{\lambda}$, como se

$a^i.x \leq b_i$ pertencesse ao conjunto original de restrições $A.x = b$ em IP:

$$DWM'' = \begin{cases} Z_{DWM''} = \min (c.Q).\lambda \\ \text{sujeito a} \\ \quad (A.Q).\lambda = b \\ \quad (a^i.Q).\lambda \leq b_i \\ \quad \mathtt{1}.\lambda = 1 \\ \quad \lambda \geq 0\,. \end{cases}$$

A adição da linha a^i à matriz A e da variável dual μ_i a μ não altera a geração de colunas. Contudo, esses dois elementos devem agora ser considerados no cálculo de $(c - \mu.A)$ na resolução do problema PS.

2.2 Resolução de *IPs*

A teoria de programação linear foi proposta na década de 40 e logo foi observado que seria desejável a resolução de problemas que apresentavam variáveis do tipo inteiro [29]. Isto levou ao desenvolvimento dos primeiros algoritmos que utilizam a idéia de se adicionar novas desigualdades à formulação do problema, os chamados *cutting plane algorithms*, propostos por Dantzig, Fulkerson e Johnson [26] e Gomory [42, 43, 44]. Em seqüência, Land e Doig introduziram em 1960 um algoritmo enumerativo que utiliza estimativas no valor da solução ótima para evitar a enumeração de todas as soluções possíveis (algoritmo *branch-and-bound* [51]). A partir de então abordagens como enumeração implícita ([15]), decomposição (Benders [18]), relaxação lagrangeana (Geoffrion [39]) e diversas heurísticas (veja Zanakis e Evans [82]) foram propostas para a resolução de *IPs*.

Infelizmente, hoje após mais de 50 anos de pesquisas nessa área, nenhum dos algoritmos disponíveis mostrou-se adequado para resolver satisfatoriamente muitas das instâncias práticas de problemas de programação linear inteira. Atualmente, boa parte dos esforços de pesquisa concentram-se no desenvolvimento de técnicas e algoritmos para classes particulares de problemas. Muitos desses algoritmos são variações do algoritmo *branch-and-bound*, descrito na seção 2.2.1, adaptado a algum problema em particular.

Nesta seção, que trata da resolução de *IPs* por algoritmos enumerativos, inicialmente é descrito na seção 2.2.1 o algoritmo *branch-and-bound* propriamente dito. Nas seções 2.2.2, 2.2.3 e 2.2.4 são apresentadas breves descrições gerais das variações *branch-and-cut*, *branch-and-price* e *branch-*

and-cut-and-price, respectivamente. Descrições mais detalhadas dessas variações podem ser encontradas nas referências [67, 50, 80, 61, 16], que foram a base para o texto dessas seções.

Para simplificar as descrições do algoritmo *branch-and-bound* e de suas variações, estas serão concentradas em formulações de programação linear inteira que não contém variáveis contínuas, ou seja, em *IPs*. Seja P um tal problema de programação linear inteira definido por 2-2. Essas descrições utilizam, como estimativa do valor da solução ótima de P, o valor da solução ótima de um problema de programação linear, habitualmente referido por relaxação linear de P e definido como $\overline{z} = min\{f(x) = c.x \mid x \in \overline{\mathcal{F}} \supseteq \mathcal{F}\}$, onde

$$\overline{\mathcal{F}} = \{x \in \mathbb{R}^n_+ \mid A.x \geq b\}\,. \tag{2-6}$$

2.2.1 *Branch-and-Bound*

O método *Branch-and-bound*, proposto por Land and Doig [51], para a resolução de problemas de programação inteira utiliza uma estratégia de divisão e conquista. O seu princípio básico é o uso de estimativas no valor da solução ótima de um problema, para evitar a inspeção de partes de seu conjunto de soluções.

Resolver um problema NP-Difícil na otimalidade geralmente é um trabalho enorme, o que requer um algoritmo eficiente e o algoritmo de *branch-and-bound* é uma das principais ferramentas para a construção desses algoritmos. O algoritmo de *branch-and-bound* procura por todo o espaço de soluções, pela melhor solução de um dado problema. Todavia, a enumeração explicita de um problema é geralmente impossível devido ao crescimento exponencial do número de soluções. O uso de limites inferiores e superiores em conjunto com o valor da melhor solução encontrada até o momento, permite ao algoritmo procurar implicitamente por apenas algumas partes do espaço de busca.

O algoritmo de *branch-and-bound* pode ser enunciado em termos simples. Uma árvore enumerativa de problemas lineares contínuos é formada, na qual cada problema tem as mesmas restrições e função objetivo do problema original P, exceto por alguns limites impostos em algumas componentes do problema. Na raiz da árvore está o problema P sem a restrições de integralidade. A solução no nó raiz não tem, em geral, todas as componentes inteiras. Escolhemos então uma componente fracionária da solução x_j e particionamos o conjunto de soluções em dois através da imposição

das restrições $x_j \leq \lfloor \overline{x}_j \rfloor$, para o primeiro, e $x_j \geq \lceil \overline{x}_j \rceil$, para o segundo. O processo de *branching* pode ser executado recursivamente, cada um dos dois novos problemas vão gerar dois novos problemas quando é feito o *branch* em uma componente fracionária de suas soluções.

Finalmente, após colocar um número suficiente de limites nas variáveis, encontramos soluções inteiras. O valor da melhor solução inteira obtida até o momento é guardado e usado como base para podar a árvore. Se o problema linear de um dos nós da árvore tem o valor da função objetivo maior do que a melhor solução inteira conhecida, todos os seus descendentes também terão, uma vez que possuem regiões viáveis menores e, desta forma, valores de função objetivo maiores. O *branch* oriundo deste nó não pode alcançar uma solução inteira melhor do que a já obtida até o momento, então não o consideramos mais, ou seja, ele é podado.

A cada iteração do *branch-and-bound* temos que decidir duas componentes principais do algoritmo: a escolha do nó a ser processado (estratégia de busca) e a escolha da variável fracionária na qual será feito o *branching*. Quanto à estratégia de busca, as mais conhecidas são: a busca em profundidade (*depth-first*) e a busca em largura (*breadth-first*). A estratégia de busca em profundidade escolhe um dos dois novos problemas gerados para ser o próximo a ser resolvido. A razão para utilizar esta estratégia é que, na prática, as soluções ótimas são encontradas em regiões profundas da árvore. Já a estratégia de busca em largura sempre escolhe o nó mais antigo ainda não explorado para ser o próximo nó a ser processado.

Em qualquer instante do processo, o status da solução, com respeito à busca no espaço de soluções, é descrito pelos ramos ainda não explorados e pela melhor solução conhecida até o momento.

2.2.2 *Branch-and-Cut*

O algoritmo *branch-and-bound* (veja seção 2.2.1) pode ser refinado para, dado um problema P de programação inteira (definido por 2-2), resolver relaxações lineares correspondentes a formulações de P com um número grande (potencialmente exponencial) de restrições. Tais formulações são de grande interesse devido ao fato de serem as únicas possíveis para determinados problemas ou, em outros casos, fornecerem melhores estimativas para o valor da solução ótima do problema do que formulações com um número polinomial de restrições. Este refinamento é conhecido como *branch-and-cut* [64].

Para obter sucesso na resolução de problemas de programação inteira é necessário uma formulação cuja relaxação linear do problema forneça uma boa aproximação da casca convexa das soluções viáveis. Nas últimas décadas, deu-se muita atenção ao *branch-and-cut* para resolver tais problemas.

A idéia básica do *branch-and-cut* é simples. Classes de desigualdades válidas, de preferência facetas da casca convexa, são deixadas de fora da relaxação linear do problema, pois existem diversas restrições e muitas delas não estarão ativas na solução ótima. Então, resolve-se o problema linear. Se a solução ótima do problema linear é inviável para o problema original, um subproblema chamado problema de separação, tenta identificar desigualdades válidas violadas por tal solução. Se este encontrar uma ou mais restrições, estas são adicionadas ao problema para cortar a solução inviável. O problema linear é então reotimizado. Este processo continua até que nenhuma desigualdade seja encontrada para cortar a solução inviável, então é feito o *branch*.

2.2.3 *Branch-and-Price*

Analogamente ao descrito na seção 2.2.2, o algoritmo *branch-and-bound* (veja seção 2.2.1) também pode ser adaptado para, dado um problema de programação inteira (definido por 2-2), resolver relaxações lineares deste problema com um elevado número (potencialmente exponencial) de variáveis. Este refinamento é conhecido como *branch-and-price*.

A idéia do *branch-and-price* é similar a do *branch-and-cut*, a diferença é que enquanto o *branch-and-cut* se concentra na geração de desigualdades válidas, o *branch-and-price* se concentra na geração de colunas. No *branch-and-price* as colunas são deixadas de fora da relaxação linear pois existem muitas e grande parte delas terão a variável associada igual a zero na solução ótima. Então, para checar a otimalidade da solução, um sub-problema, chamado de *pricing problem*, é chamado para tentar identificar colunas com custo reduzido negativo. Se alguma coluna for encontrada, o problema é então re-otimizado. O *branch* ocorre quando o sub-problema não encontra nenhuma coluna de custo reduzido negativo e a solução da relaxação linear não satisfaz a condição de integralidade.

Considere o problema de programação linear inteira IP, como formulado na seção 2.1.3. Aplicando-se a IP a reformulação tradicional para problemas de programação linear inteira, descrita na seção 2.1.3, obtém-se o chamado problema mestre IP'. O problema DWM (*Dantzig-Wolfe Mas-*

ter), definido na seção 2.1.1, é obtido relaxando-se as restrições de integralidade em IP'. Um algoritmo *branch-and-price* para resolução do problema IP trabalha em cada nó da árvore de enumeração com uma versão restrita $DWMR$ de DWM, em que apenas um subconjunto de colunas é mantido na formulação corrente, conforme o esquema de geração de colunas descrito na seção 2.1.2.

Relembre-se aqui a necessidade de resolver até a otimalidade o DWM para cada subproblema criado, caso contrário a estimativa obtida pode não ser válida. Alternativamente, pode-se utilizar a estimativa lagrangeana [76] que produz uma estimativa inferior válida, cada vez que o subproblema de geração de colunas é resolvido até a otimalidade.

A grande dificuldade, talvez a maior, no desenvolvimento de algoritmos *branch-and-price* é o estabelecimento de critérios adequados de *branching*. Um tal critério deve ser aplicável não somente ao problema mestre, mas também ao subproblema de geração de colunas. Em conseqüência, a verificação da existência de colunas de custo reduzido negativo deve satisfazer todos os critérios de *branching* utilizados no ramo atual da árvore de enumeração, desde o nó raiz até o nó corrente. Assim, o critério utilizado pode alterar a estrutura do subproblema usado na geração de colunas e tornar essa operação cada vez mais complicada.

Diversos trabalhos ([16, 46, 69, 74, 75, 79]) exploram diferentes alternativas de *branching* em algoritmos *branch-and-price*, muitas delas específicas para problemas particulares.

2.2.4 *Branch-and-Cut-and-Price*

O desenvolvimento de algoritmos enumerativos associados a métodos de geração de cortes (algoritmos *branch-and-cut* – seção 2.2.2) e de geração de colunas (algoritmos *branch-and-price* – seção 2.2.3) é um campo razoavelmente bem explorado. Contudo, desde que a geração de cortes e a de colunas foram estabelecidas como duas das técnicas mais importantes na programação inteira, tem-se procurado maneiras de combiná-las de forma eficiente em um mesmo algoritmo.

O algoritmo *branch-and-cut-and-price* é uma especialização do *branch-and-bound* (veja seção 2.2.1) em que novas colunas e novas desigualdades válidas são geradas dinamicamente à medida que a árvore de busca é percorrida. Embora este algoritmo utilize várias das técnicas usadas nos algoritmos *branch-and-cut* e *branch-and-price* (que essencialmente tem o

mesmo princípio básico), o resultado dessa combinação requer técnicas muito mais sofisticadas do que as utilizadas em cada um em separado. Conforme descrito na seção 2.1.3, uma das razões é a necessidade de se acrescentar novas desigualdades (cortes) sem alterar a estrutura do subproblema de geração de colunas.

Em [66] Poggi de Aragão e Uchoa denominam *branch-and-cut-and-price* "robusto"a tais algoritmos em que a geração de novos nós na árvore de enumeração ou a adição de novas desigualdades não muda a estrutura dos subproblemas usados na geração de colunas. Dado que uma operação de *branching* pode ser vista como uma inserção de cortes nos vários subproblemas da árvore de enumeração, a mesma técnica usada na adição de cortes pode ser usada para *branching*.

Os bons resultados dos experimentos computacionais obtidos com a aplicação de algoritmos *branch-and-cut-and-price* robustos a problemas como *Cap-MST* (*Capacitated Minimum Spanning Tree*) [35], *CVRP* (*Capacitated Vehicle Routing*) [36] e *GAP* (*Generalized Assignment Problem*) [65] mostram o poder da combinação da geração de cortes e colunas.