Branch-and-bound

Prof. Alexandre Salles da Cunha

Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Belo Horizonte, Brasil

acunha@dcc.ufmg.br

Setembro 2020

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

UFMG - ICEx
DEPARTAMENTO DE CIÊNCIA DA
COMPUTAÇÃO

Relativas à esta apresentação

1. L. Wolsey. Integer Programming, Wiley, 1998. [Capítulo 7]
2. D. Bertsimas e J. N. Tsitsiklis. Introduction to Linear Optimization, Athena Scientific, 1997. [Seção 11.2]
3. R. J. Vanderbei. Linear Programming: Foundations and Extensions, Springer, 4a. Edição, 2014. [Seção 23.5]

Dado o problema

$$z = \max\{c^T x : x \in S\}$$

como podemos fazer para dividi-lo em um conjunto de problemas menores, possivelmente mais fáceis de serem resolvidos e reunir a informação obtida com a solução de cada um dos subproblemas de forma a resolver o problema original ?

Proposição

Seja $S = S_1 \cup \cdots \cup S_K$ a decomposição de S em K subconjuntos menores e seja $z^k = max\{c^T x : x \in S_k\}, k = 1, \ldots, K$. Então $z = max_{k=1,\ldots,K}\{z^k\}$.

Observe que esta idéia é geral, não restrita à Programação Linear Inteira (ou inteira mista).

Um modo trivial de decompor o problema

Uma maneira de representar a abordagem de Dividir e Conquistar é através de uma árvore de enumeração. No caso onde, por exemplo, $S = \mathbb{B}^3$ podemos subdividir através de uma árvore de enumeração binária.

- Na árvore de enumeração binária anterior existem 3 níveis, cada qual associado a uma decisão de fixar uma variável em 0 ou em 1.
- Cada folha da árvore de enumeração corresponde a uma solução que deveria ser avaliada, caso se deseje resolver um problema de otimização definido em S, através de enumeração completa.
- Nesta árvore existem 2^n $(n = 3)$ folhas.

Decompondo o espaço de soluções do TSP em 4 cidades

Considere que S denota o conjunto de tours em 4 cidades. Uma maneira de decompor S é dado pela árvore abaixo.

- S_{ij} denota o conjunto de tours que selecionam a aresta (i, j).
- Existem $(n-1)!$ folhas na árvore de enumeração.

Outra ideia importante: Enumeração Implícita

- A enumeração total do espaço de soluções (desenvolvendo a árvore de enumeração até que totas as suas folhas sejam avaliadas) é impraticável para problemas de dimensões de interesse.
- Precisamos fazer algo melhor do que simplesmente enumerar o espaço de soluções exaustivamente.

Uso de limites para podar a árvore

A idéia central consiste em empregar limitantes inferiores e superiores para o valor ótimo z^k do problema S_k, de modo que não seja necessário desenvolver a árvore de enumeração abaixo do nó que representa S_k.

Proposição

Seja $S = S_1 \cup \cdots \cup S_K$ a decomposição de S em subconjuntos menores e seja $z^k = max\{c^T x : x \in S_k\}$ para $k = 1, \ldots, K$. Seja $\overline{z}^k$ um limite superior para z^k e $\underline{z}^k$ um limite inferior para z^k. Então, $\overline{z} = max_k\{\overline{z}^k\}$ é um limite superior para z e $\underline{z} = max_k\{\underline{z}_k\}$ é um limite inferior para z.

A enumeração é usualmente organizada na forma de uma lista

E a lista representa as folhas de uma árvore, os subproblemas ainda não investigados....

O que pode ser dito sobre z considerando os limites superiores e inferiores indicados em cada nó ? Considere uma *fotografia da árvore* dada pela exploração de 3 subproblemas, S, S_1, S_2, isto é, S foi decomposto em 2 subproblemas. Para cada um destes subproblemas, limites inferiores e superiores foram avaliados.

Poda da árvore por limites

Considere a seguinte *fotografia da árvore* dada pela exploração de 3 subproblemas, S, S_1, S_2.

Desejamos resolver $\max\{c^T x : x \in S\}$.

- Suponha que disponhamos de uma solução viável que produz um limite inferior (primal) $\underline{z}$ válido para o valor ótimo z, isto é, $z \geq \underline{z}$.
- Suponha agora ao invés de resolver o problema k da árvore de enumeração, encontramos um limite superior (dual) $\overline{z}^k$ para o valor de $z^k = \max\{c^T x : x \in S_k\}$, isto é, $z^k \leq \overline{z}^k$.
- O que podemos afirmar se $\overline{z}^k \leq \underline{z}$?

Poda da árvore por limites duais

- Vamos analisar o espaço de soluções de um descendente de S_k, digamos S_p.
- Claramente temos que $S_p \subset S_k$, isto é, S_p é mais restrito que S_k. Assim sendo $z^p \leq z^k$. Então $z^p \leq z^k \leq \overline{z}^k < \underline{z}$.
- Ou seja, neste caso, não precisamos desenvolver a árvore de enumeração completamente a partir de S_k.

Impossibilidade de poda, necessidade de branching

Considere a seguinte *fotografia da árvore* dada pela exploração de 3 subproblemas, S, S_1, S_2.

Resumindo

1. Por invialibidade. Por exemplo: Se a relaxação é inviável, o IP representado pelo subproblema é inviável.
2. Por otimalidade: se o subproblema foi resolvido à otimalidade.
3. Por limites (maximização): o limite inferior (valor da função objetivo da melhor solução viável conhecida) iguala ou excede o melhor limite dual associado ao subproblema.

Combinando as duas ideias

O método Branch-and-bound consiste então na enumeração inteligente do espaço de soluções de um Problema de Otimização Inteira. O termo *Branch* diz respeito à partição do domínio enquanto que o termo *Bound* diz respeito ao uso de limites primais e duais para construção de uma prova de otimalidade de forma a evitar a busca exaustiva.

1. Qualquer problema de otimização em que possamos particionar o domínio e produzir limites primais e duais é um problema que pode ser resolvido através de um algoritmo do tipo Branch-and-bound.
2. Assim sendo, o Branch-and-bound não é um algoritmo específico, mas sim uma classe geral de algoritmos.

Algumas questões pertinentes

1. Como escolher relaxações ?
2. Devemos gerar limites duais *mais fracos* porém *mais baratos* de serem avaliados computacionalmente ou devemos nos concentrar em limites mais fortes e mais caros ? Como equilibrar estes aspectos ?
3. Como devemos estabelecer a decomposição $S = S_1 \cup \ldots S_K$? Idealmente estes conjuntos devem ser disjuntos. Este princípio é no entanto pouco restritivo. O critério de decomposição deve ser fixado a priori ou deve mudar na medida em que novos limites e soluções viáveis são encontrados ao longo da enumeração ?
4. Em qual ordem os subproblemas na lista, isto é, os subproblemas ainda não resolvidos, devem ser escolhidos para investigação ?

PI

$$
\begin{array}{lcrcrcl}
z & = & \max & 4x_1 & - & x_2 & \\
 & & & 7x_1 & - & 2x_2 & \leq & 14 \\
 & & & & & x_2 & \leq & 3 \\
 & & & 2x_1 & - & 2x_2 & \leq & 3 \\
 & & & x_1, & & x_2 & \in & \mathbb{Z}_+
\end{array}
$$

Introduzimos variáveis de folga e resolvemos a Relaxação Linear.

Base ótima $B^* = \{1, 2, 5\}$

$$
\begin{array}{lllllllll}
\bar{z} & = & \max \frac{59}{7} & & & -\frac{4}{7}x_3 & -\frac{1}{7}x_4 & & \\
 & & & x_1 & & + \frac{1}{7}x_3 & + \frac{2}{7}x_4 & = & \frac{20}{7} \\
 & & & & x_2 & & + x_4 & = & 3 \\
 & & & & & -\frac{2}{7}x_3 & + \frac{10}{7}x_4 + x_5 & = & \frac{23}{7} \\
 & & & x_1, & x_2, & x_3, & x_4, & x_5 \in & \mathbb{Z}_+
\end{array}
$$

- A Relaxação Linear nos fornece $\bar{z} = \frac{59}{7}$.
- Assumindo que não dispomos de nenhuma solução viável para o problema, fazemos $\underline{z} = -\infty$.
- Uma vez que $\underline{z} < \bar{z}$, devemos particionar o problema.

Uma opção: Branching em Variáveis

Dado $x_j \in \mathbb{Z}$ que na relaxação linear é variável básica e valor assume $\overline{x}_j \notin \mathbb{Z}$, particionamos S da seguinte forma:

$$S_1 = S \cap \{x : x_j \leq \lfloor \overline{x}_j \rfloor\}$$

$$S_2 = S \cap \{x : x_j \geq \lceil \overline{x}_j \rceil\}$$

Observe que:

- $S = S_1 \cup S_2$ (condição necessária para garantia de otimalidade);
- Neste caso, $S_1 \cap S_2 = \emptyset$ (economia);
- A solução ótima $\overline{x}$ da relaxação linear viola as restrições que determinam o particionamento do domínio. Limites duais mais fortes devem ser encontrados por meio da resolução das Relaxações Lineares dos subproblemas S_1 e S_2.

Mais um grau de liberdade

Quando a solução da Relaxação Linear possui mais de uma componente fracionária que deveria ser inteira, em qual variável devemos implementar o Branching ?

Uma vez que $\overline{x}_1 = \frac{20}{7} \notin \mathbb{Z}$, fazemos

$$S_1 = S \cap \{x : x_1 \leq 2\}$$

$$S_2 = S \cap \{x : x_1 \geq 3\}.$$

Qual nó em aberto devemos escolher para resolver primeiro ?

Critério de Seleção: escolha do nó da lista para explorar

Via de regra, podemos adotar estratégias que dão prioridade a:

1. Obtenção de soluções viáveis mais rapidamente (busca em profunidade - escolhemos o nó mais distante da raiz)
2. Prova de Otimalidade: escolhemos o nó com a pior estimativa de limite dual, maior $\overline{z}^k$ (caso obtenhamos uma solução viável através deste nó, melhoramos muito o limite primal)
3. Um equilíbrio das duas alterantivas (primeiro em profundidade, depois em limite).

Após escolhido o nó, reotimizamos

- Isto feito, vamos reotimizar a partir do nó pai. Como podemos poupar tempo para resolver a relaxação linear associada a S_2, uma vez que já dispomos de $\overline{x}$, solução para a relaxação linear de S ?

- É natural efetuar esta reotimização através do Método Dual Simplex, uma vez que a inserção da restrição $x_1 \leq 2$ não altera a viabilidade do dual. Isto implica em fazer um número significativamente inferior de operações de pivoteamento (inversões de matrizes) durante o Método Simplex.

Vamos escolher abritrariamente o nó S_1 para explorarmos. Inserimos então no modelo a restrição $x_1 \leq 2$ ou $x_1 + s = 2, s \geq 0$:

$$\frac{20}{7} - \frac{1}{7}x_3 - \frac{2}{7}x_4 + s = 2 \text{ ou}$$

$$-\frac{1}{7}x_3 - \frac{2}{7}x_4 + s = -\frac{6}{7}$$

Então temos o programa linear a resolver, via Dual Simplex:

Calculando $LP(S_1)$

$$\begin{array}{lllllllllll}
\overline{z}_1 & = & \max \frac{59}{7} & & & -\frac{4}{7}x_3 & -\frac{1}{7}x_4 & & & & \\
& & & x_1 & & + \frac{1}{7}x_3 & + \frac{2}{7}x_4 & & & = & \frac{20}{7} \\
& & & & x_2 & & + x_4 & & & = & 3 \\
& & & & & -\frac{2}{7}x_3 & + \frac{10}{7}x_4 & + x_5 & & = & \frac{23}{7} \\
& & & & & -\frac{1}{7}x_3 & -\frac{2}{7}x_4 & & +s & = & -\frac{6}{7} \\
& & & x_1, & x_2, & x_3, & x_4, & x_5 & s & \in & \mathbb{Z}_+
\end{array}$$

Resolução de $LP(S_1)$ com duas operações de pivoteamento

$LP(S_1)$ resolvida

$$
\begin{array}{lcl}
\overline{z}_1 & = & \max \frac{15}{2} - \frac{1}{2}x_5 - 3s \\
x_1 & & + s = 2 \\
x_2 & & - \frac{1}{2}x_5 + s = \frac{1}{2} \\
x_3 & & - x_5 - 5s = 1 \\
x_4 & & + \frac{1}{2}x_5 + 6s = \frac{5}{2}
\end{array}
$$

- Os limites inferior e superior globais não podem ser atualizados;
- S_1 não pode ser podado, então criamos dois novos nós:
  - $S_{11} = S_1 \cap \{x : x_2 \leq 0\}$
  - $S_{12} = S_1 \cap \{x : x_2 \geq 1\}$

Escolhendo o próximo nó e reotimizando

Dentre os nós ativos $\{S_2, S_{11}, S_{12}\}$, arbitrariamente escolhemos S_2 para explorar.

Resolvendo $LP(S_2)$

- Combinando a primeira restrição da Relaxação Linear do modelo com $x_1 \geq 3$ temos: $2x_2 \geq 7x_1 - 14 \geq 7$. Esta desigualdade é incompatível com a restrição $x_2 \leq 3$.
- Logo, verificamos que $LP(S_2)$ é inviável.
- Então $\overline{z}_2 = -\infty$ e o nó é podado por inviabilidade.
- Os melhores limites inferior e superior globais não podem ser atualizados.

Escolhendo o próximo nó

Dentre os nós ativos $\{S_{11}, S_{12}\}$, arbitrariamente escolhemos S_{12} para explorar.

Avaliando $LP(S_{12})$

- A solução de $LP(S_{12})$ é inteira: $x^{12} = (2,\ 1)^T$
- Neste caso $\bar{z}_{12} = 7$ fornece um limite inferior (primal) válido, atualizamos $\underline{z}$: $\underline{z} \leftarrow \max\{\underline{z}, 7\} = \max\{-\infty, 7\} = 7$.
- O nó S_{12} pode ser podado por otimalidade.

Escolhemos o único nó restante, S_{11}, para explorar

- A relaxação linear ótima é $x = (\frac{3}{2},\ 0)^T$ e a função objetivo correspondente é $\overline{z} = 6$. Como $\overline{z} = 6 < \underline{z} = 7$, podamos o nó por limites (bounds).
- A lista de subproblemas a explorar está vazia.
- Logo $z = 7$ é o valor ótimo da função objetivo e uma solução ótima é dada por $(2,\ 1)^T$.

Aspectos de Implementação de algoritmos BB baseados em LP

1. O que deve ser armazenado.
2. Como reduzir o tamanho da árvore de enumeração.
3. Uso dos algoritmos de Programação Linear.
4. Como realizar:
   - Branching.
   - Seleção de nós.
5. Paralelização.

Armazenamento

- Na prática, não se armazena a árvore, mas sim uma lista de nós ativos (cuja investigação futura ainda não foi ser descartada).
- Em cada nó ativo, necessariamente precisamos guardar:
  - O limite dual referente ao pai do nó.
  - Os limites inferiores e superiores associados a cada variável (que podem ser restringidos ao longo da árvore)
- Em algoritmos baseados em PL, usualmente armazena-se também a base ótima associada ao nó pai ou uma base avançada.
- Em algoritmos baseados em Relaxação Lagrangeana, é usual armazenar os melhores multiplicadores de Lagrange obtidos na resolução do nó pai.

O aspecto central

A redução rápida do gap de dualidade $\frac{\overline{z}-\underline{z}}{\underline{z}}$.

Como fazer isto:

- Aspectos ligados à representação do problema em estudo:
  - Adotando formulações mais fortes.
  - Empregando cortes (Branch-and-cut) para fortalecer as formulações.
  - Convexificando parte do domínio (Geração de colunas, Branch-and-price, Relaxação Lagrangeana)
  - Pré-processamento do problema
- Obtendo soluções viáveis de melhor qualidade, por exemplo, encasulando o uso de heurísticas primais ao longo da árvore (baseadas em informação de PL).
- Escolhas de implementação: uso de *boas* regras de seleção e de Branching.

Regras de Branching

- Branching em variáveis: fáceis de implementar, comuns na maioria dos pacotes comerciais. Entretanto, podem gerar árvores desbalanceadas e via de regra é desejável estabelecer prioridade de branching entre as variáveis.
- Branching em restrições: mais específicas, dependentes do problema e assim sendo, podem produzir resultados muito melhores para alguns problemas específicos.

Critério simples - privilegia violação da integralidade

Escolha a variável inteira cuja parte fracionária na relaxação linear é mais próxima de $\frac{1}{2}$.

Um critério um pouco mais sofisticado e mais caro

Estimar a deterioração do limite dual ocasionada pela fixação de cada variável candidata a branching nos limites que definem o branching. Esta deterioração pode ser feita:

- De forma exata, implementando tantas quantas operações de pivoteamento (dual) quanto forem necessárias.
- De forma aproximada, seja estimando a degradação atavés do preço (variável dual) associado à restrições: $x_j \leq u_j$ e $x_j \geq l_j$ ou implementando algumas poucas operações de pivoteamento.

Definição

Consiste em efetuar a avaliação da degradação de forma exata, realizando tantas quantas operações de pivoteamento no dual Simplex quanto forem necessárias, para um conjunto restrito C de variáveis candidatas a branching.

Implementando

Se C denota o conjunto restrito de variáveis básicas candidatas, fazemos o branching na variável

$$j^* \in \arg\min_{j \in C}\{\max\{\overline{z}_j^D, \overline{z}_j^U\}\},$$

onde $\overline{z}_j^D$ ($\overline{z}_j^U$) denota o limite dual obtido após impor $x_j \leq \lfloor \overline{x_j} \rfloor$ ($x_j \geq \lceil \overline{x_j} \rceil$)

Uncapacitated Facility Location

$$\min \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{j \in N} f_j y_j$$

$$\sum_{j=1}^{n} x_{ij} = 1, \; i = 1, \ldots, m$$

$$x_{ij} \leq y_j, \; i = 1, \ldots, m, \; j = 1, \ldots, n$$

$$x \in \mathbb{B}^{n \times m}, y \in \mathbb{B}^n$$

Questão

Todas as variáveis binárias possuem a mesma relevância para o branching ?

Variáveis com papéis muito diferentes

Observe que fixar uma facilidade $j \in N$ (ou y_j) em 0 ou em 1 altera muito mais a natureza da solução que, dado $j : y_j \neq 0$, fixar $x_{ij} = 0$ ou $x_{ij} = 1$ para algum $i \in M$.

Alternativa

Assim sendo, muitos pacotes de Programação Inteira permitem estabelecer prioridades de Branching para as variáveis inteiras. No caso do UFL, as variáveis binárias y possuem maior prioridade de branching que as variáveis x.

Considere o problema do caixeiro viajante e as desigualdades

$$x(\delta(U)) \geq 2k, \ \forall U \subseteq V, \text{ para algum } k \in \mathbb{Z}_+.$$

Suponha que $\overline{x}(\delta(U)) = 2.5$ para algum U.
Podemos então particionar o domínio criando dois nós:

- $x(\delta(U)) = 2$
- $x(\delta(U)) \geq 4$

Seleção de nós

Dilema: Soluções viáveis mais rápido vs melhorar o limite dual.

- Por um lado, o número de nós ativos cresce mais rapidamente se o nó a ser explorado é sempre escolhido dentre aqueles mais próximos da raiz da árvore. Tipicamente, os limites duais melhoram mais rapidamente neste caso.
- Por outro lado, ao escolher nós mais distantes da raiz, os limites duais globais melhoram muito pouco. Esta estratégia permite encontrar soluções viáveis mais rapidamente.

Uma solução de compromisso

Primeiro implementar uma busca em produndidade, procurando uma solução viável inicial. Em seguida, alternar busca em largura com busca em profundidade.