Combinatória Poliédrica
Branch-and-Cut

Rafael C. S. Schouery
rafael@ic.unicamp.br

Universidade Estadual de Campinas

Atualizado em: 2023-11-15 13:34

Branch-and-Cut

Veremos uma implementação do método *Branch-and-Cut*

- Para resolver um problema de programação linear inteira

Problema de Steiner em Grafos: Dado um grafo $G = (V, A)$, um conjunto de vértices $Z \subseteq V$ chamados terminais e uma função de custo $c\colon A \to \mathbb{R}$, encontrar $S \subseteq A$ tal que $G[S]$ conecta todos os terminais e $c(S)$ seja mínimo.

$S \subseteq V$ tal que $G[S]$ conecta todos os terminais é chamado de uma solução de Steiner

Formulação de Programação Linear Inteira

$$P_{\text{Ste}}(G, Z) = \text{conv}\{\boldsymbol{\chi}^S : S \text{ é uma solução de Steiner}\}$$

Formulação de Programação Linear Inteira:

$$\begin{aligned} \min \sum_{e \in A} c_e x_e & \\ x(\delta(W)) \geq 1 & \qquad \forall W \subseteq V, \emptyset \neq W \cap Z \neq Z \\ x_e \in \{0, 1\} & \qquad \forall e \in A \end{aligned}$$

Branch-and-Bound

A ideia do Branch-and-Bound é enumerar as soluções viáveis

- Usando uma árvore de decisão
- Como em um backtracking
- Mas podando os ramos que não levam a soluções ótimas
- E não necessariamente percorrendo em profundidade

No caso do Problema de Steiner, podemos, para uma aresta $e \in A$

- Considerar o subproblema em que $x_e = 1$
- E o subproblema em que $x_e = 0$
- Essa é a parte "Branch" do método

Assím podemos enumerar as soluções em uma árvore de altura $|A|$

- Ou seja, as $2^{|A|}$ possíveis soluções
- Nem todas viáveis...

Não parece uma boa ideia do jeito que está...

Limitantes

Podemos melhorar a enumeração usando limitantes superiores e inferiores para o valor da solução ótima

- Essa é a parte "Bound" do método

Se soubermos um valor de alguma solução viável para o problema, temos um limitante superior para o valor da solução ótima

- Chamamos essa solução de incumbente

Se uma subárvore tem apenas soluções piores ou iguais ao incumbente, podemos podar essa subárvore

- Mas como saber isso?
- Podemos usar um limitante inferior para o valor da solução ótima da subárvore

Relaxação Linear: Limitantes, Viabilidade e Incumbentes

A Relaxação Linear do subproblema é útil para o B&B pois

- Se a Relaxação Linear do subproblema é inviável, então o subproblema é inviável
  - Consigo enumerar menos soluções
- Se a Relação Linear tem solução ótima inteira, então essa solução é ótima para o subproblema
  - Potencialmente nos dá um novo incumbente
  - Não é necessário continuar enumerando
- E o valor da Relaxação Linear é um limitante inferior para o valor da solução ótima do subproblema
  - Permite podar a subárvore se o limitante for maior ou igual ao incumbente

Branch-and-Bound (com Programação Linear)

Inicialização:

1. Tente calcular uma solução ótima x^* da Relaxação Linear do Problema
   a. Se é inviável, devolva que o problema é inviável
   b. Se é ilimitada, devolva que o problema é ilimitado ou inviável
   c. Se a solução encontrada é inteira, devolva essa solução
2. Armazene o problema e x^* em uma estrutura de dados
   – Ex: fila, pilha, fila de prioridade por valor da solução
3. Inicialize o incumbente com vazio e valor $+\infty$

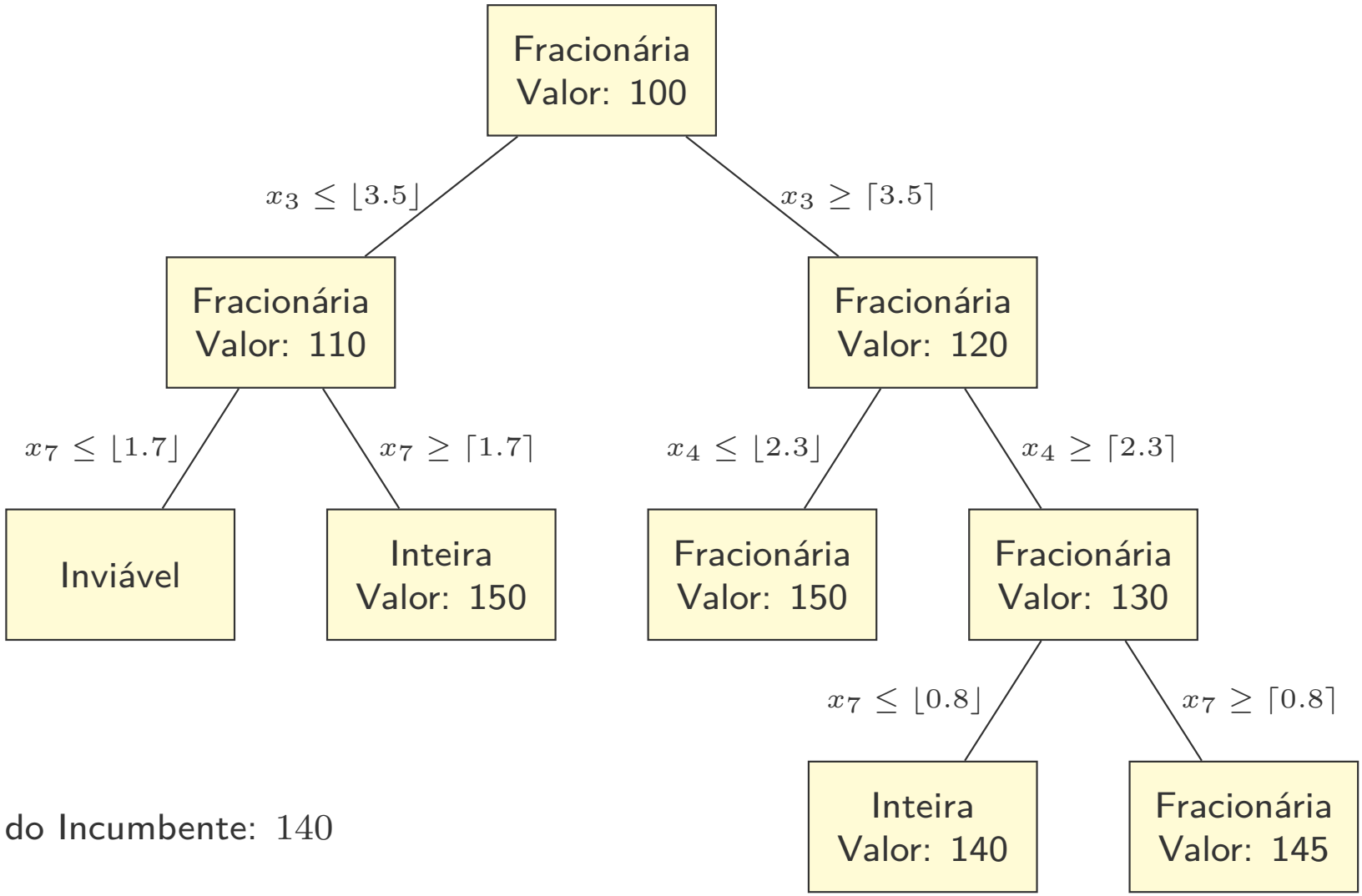
Branch-and-Bound (com Programação Linear)

Enquanto a estrutura de dados não está vazia:

1. Retire um problema P e a solução fracionária ótima x^* da estrutura de dados
2. Se o valor de x^* é maior ou igual ao valor do incumbente, vá para a próxima iteração
3. Escolha i tal que x^*_i é fracionário
4. Crie um problema P_1 com a restrição adicional $x_i \leq \lfloor x^*_i \rfloor$
5. Crie um problema P_2 com a restrição adicional $x_i \geq \lceil x^*_i \rceil$
6. Para $j = 1, 2$, tente calcular a Relaxação Linear de P_j
   - Se é inviável, vá para a próxima iteração
   - Seja x^* a solução ótima da Relaxação Linear de P_j
   - Se x^* é inteira, atualize o incumbente (se necessário) e vá para a próxima iteração
   - Caso contrário, armazene P_j e x^* na estrutura de dados

Ao final, devolva o incumbente

Simulação



Valor do Incumbente: 140

Voltado ao Problema de Steiner

Para resolver o problema de Steiner ainda temos vários desafios

Começando pelo primeiro que é calcular a relaxação linear

$$\begin{aligned} \min \sum_{e \in A} c_e x_e & \\ x(\delta(W)) \geq 1 & \qquad \forall W \subseteq V, \emptyset \neq W \cap Z \neq Z \\ 0 \leq x_e \leq 1 & \qquad \forall e \in A \end{aligned}$$

Temos um número exponencial de restrições

Separando

Vamos separar as restrições conforme necessário

Em cada nó do Branch-and-Bound:

1. Calculamos uma solução ótima da relaxação linear
2. Verificamos se existe restrição violada
   - 2.1 Se houver restrição violada, adicionamos ela ao modelo e voltamos ao passo 1
   - 2.2 Caso contrário, paramos com a solução ótima da relaxação

Note que podemos ter que separar mesmo se a solução for inteira!

- Ela pode não ser viável para o problema

Iniciamos com algumas restrições

- É particularmente útil começar com algo limitado!
- Começamos com $0 \leq x_e \leq 1$ para todo $e \in A$
- Podemos começar com algumas restrições de Steiner também
  - Ex: todos os W com $|W| = 1$

Separando no Problema de Steiner

Dada uma solução fracionária x

- queremos encontrar um conjunto W tal que
- $W \subseteq V, \emptyset \neq W \cap Z \neq Z$
- e $x(\delta(W)) < 1$

Podemos enumerar $s, t \in Z$ e encontrar um (s,t)-corte mínimo em G considerando x como a capacidade das arestas

Existe um (s,t)-corte mínimo W tal que $x(\delta(W)) < 1$ se e somente se existe uma restrição violada

Dado (s,t), podemos encontrar um fluxo máximo e a partir dele um (s,t)-corte mínimo

- Mas temos $O(|Z|^2)$ pares (s,t)
- E algoritmo de fluxo é custoso...

Podemos então usar a árvore de Gomory-Hu!

Árvore de Gomory-Hu

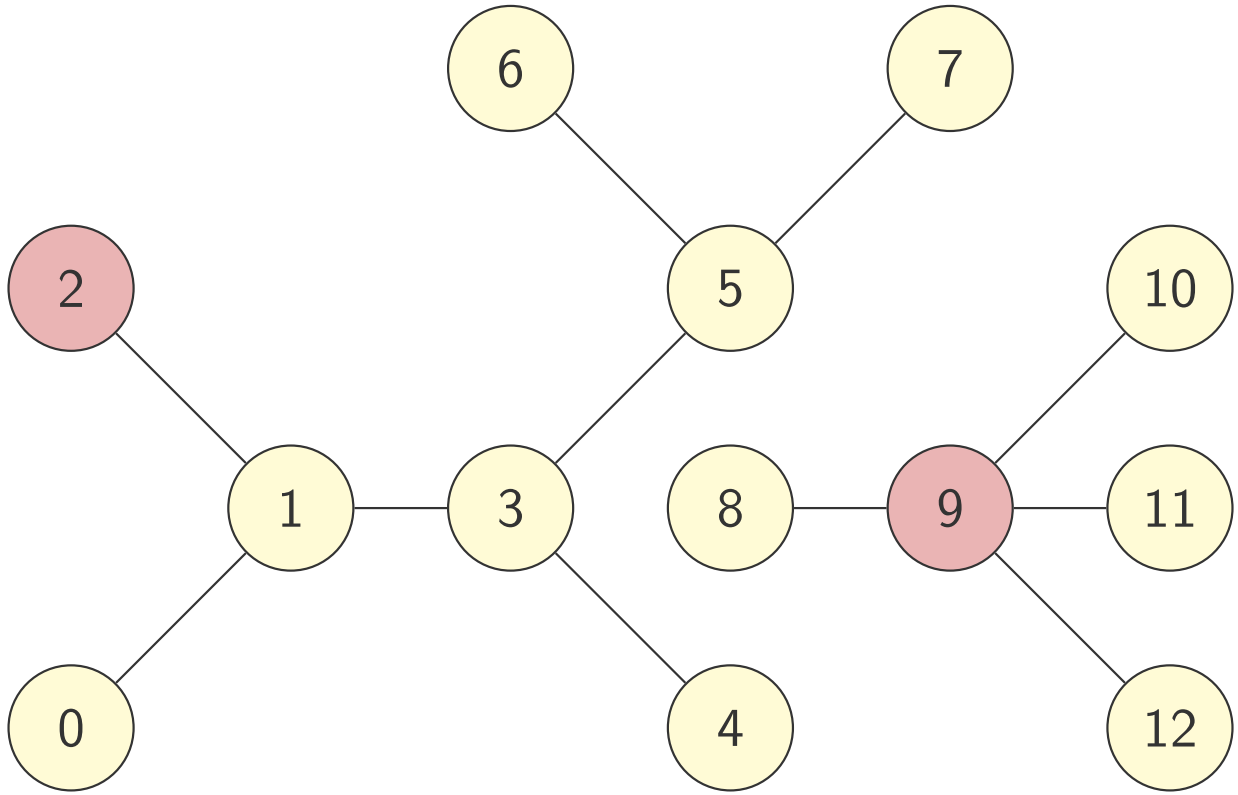
Gomory e Hu apresentaram um algoritmo que dado um grafo $G = (V, A)$ calcula uma árvore $T = (V, E)$ com pesos $c\colon E \to \mathbb{R}$ tal que

- Para todo $s, t \in V$
- Se e é uma aresta de peso mínimo no (único) caminho entre s e t em T
- Então $c(e)$ é o valor de um (s, t)-corte mínimo em G
- E $G - e$ tem duas componentes conexas que os vértices representam as partes da bipartição de um (s, t)-corte mínimo

Se existe mais de uma aresta e de peso mínimo, qualquer uma delas serve no resultado acima

O algoritmo é mais rápido do que calcular todos os cortes mínimos

Exemplo



$\delta(\{0, 1, 2, 3, 4, 5, 6, 7\})$ é um $(2, 9)$-corte mínimo

- Com peso $c(\{3, 8\})$

Melhorando a relaxação

Conseguimos então resolver a relaxação em cada nó

- Lembrando que pode ser necessário separar as restrições novamente além do nó raiz
- Mas a relaxação pode ser fraca

Para melhorar a relaxação, podemos adicionar cortes

- Idealmente que sejam facetas do poliedro

Essa é a ideia do Branch-and-Cut

- Branch-and-Bound com cortes
- Adicionados ao longo da árvore de enumeração

Relembrando... Partição de Steiner

Uma partição $\mathcal{P} = (V_1, \ldots, V_n)$ de V tal que $V_i \cap Z \neq \emptyset$ para $i = 1, \ldots, n$ é chamada de partição de Steiner

- Definimos $\Delta(\mathcal{P}) = \{\{u, v\} \in A \colon u \in V_i, v \in V_j, i \neq j\}$

Teorema. Seja G conexo e sem pontes de Steiner e tome uma partição de Steiner $\mathcal{P} = (V_1, \ldots, V_n)$ de V. Então, a inequação $x(\Delta(\mathcal{P})) \geq k - 1$ define uma faceta de $P_{\text{Ste}}(G, Z)$ se e somente se

(a) $G[V_i]$ é conexo

(b) $G[V_i]$ não contém pontes de Steiner com relação a $Z_i = Z \cap V_i$ e

(c) $\hat{G}$ obtido de G contraindo-se cada um dos conjuntos V_i a um único vértice é 2-conexo.

Generalizam as inequações de corte de Steiner

- Mas são NP-difíceis de separar...
- Podemos tentar separar com heurísticas

Heurística Gulosa

Em cada iteração temos uma partição $\mathcal{P} = (V_0, V_1, \ldots, V_k)$ tal que

- V_0 contém apenas vértices não terminais ("de Steiner")
- $V_i \cap Z \neq \emptyset$ para $i = 1, \ldots, k$

Heurística Gulosa:

1. $\mathcal{P} = (V_0, V_1, \ldots, V_{|Z|})$ em que $V_0 = V \setminus Z$ e V_i, para $i = 1, \ldots, |Z|$, são compostos de um único vértice de Z
2. Enquanto $x(\Delta(\mathcal{P})) \geq k - 1$ e $\Delta(\mathcal{P}) \neq \emptyset$:
   a. Encontre $e \in \Delta(\mathcal{P})$ que maximiza x'_e
   b. Sejam i, j tal que $i < j$ e $e = \{u, v\}$ com $u \in V_i$ e $v \in V_j$
   c. Se $i \geq 1$, então remova V_i e V_j de $\mathcal{P}$ e adicione $V_i \cup V_j$ a $\mathcal{P}$
   d. Senão, faça $V_0 = V_0 \setminus \{u\}$ e $V_j = V_j \cup \{u\}$
3. Para cada vértice u de V_0
   a. Encontre V_i que maximize $x(\{\{u, v\} : v \in V_i\})$
   b. Faça $V_i \leftarrow V_i \cup \{u\}$

A partição encontrada é uma Partição de Steiner

- Que pode (ou não) violar a restrição

Heurística de Cortes Mínimos

Heurística de Cortes Mínimos:

- $\mathcal{P} = (V_1)$ em que $V_1 = V$
- Enquanto $x(\Delta(\mathcal{P})) \geq k - 1$ e existir V_i tal que $|V_i \cap Z| \geq 2$:
  - Escolha $i \in \{1, \ldots, k\}$ tal que $|V_i \cap Z| \geq 2$
  - Encontre um corte de Steiner de capacidade mínima no grafo $G[V_i]$ usando x como capacidade das arestas
  - Seja W tal que $\delta_{G[V_i]}(W)$ é tal corte
  - Remova V_i de $\mathcal{P}$ e adicione W e $V_i \setminus W$ a $\mathcal{P}$

Ao final temos uma Partição de Steiner

- Que ou não pode mais ser quebrada
- Ou que viola a restrição

Quais cortes adicionar?

Quanto mais cortes adicionamos

- Mais forte fica a relaxação
- Mas mais lento fica calcular a Relaxação Linear

Ou seja, nem sempre é bom adicionar todos os cortes encontrados!

Dicas:

- Procure adicionar cortes de classes diferentes
- Procure adicionar cortes que estão mais distantes de serem satisfeitos
  - Analise $\alpha - a^T x'$ para um corte $a^T x \leq \alpha$ e solução ótima x' da relaxação linear
- Procure adicionar cortes que cubram todas as variáveis
  - Se focar em algumas variáveis, a solução "foge" pelas outras variáveis
- Pode ser melhor ir para o branching do que adicionar cortes
  - Ficar adicionando cortes pode levar a *tailing off* — a melhora no valor da solução é cada vez menor

Heurísticas Primais

Durante a enumeração, podemos encontrar soluções inteiras que são soluções ótimas fracionárias da Relaxação Linear

- Potencialmente atualizando o incumbente

Mas isso pode demorar muito para acontecer...

- E um bom incumbente é essencial para enumerar pouco

Podemos então usar uma heurística (primal) para encontrar uma solução inteira

- Não apenas na raiz, mas também durante a enumeração

Uma ideia é, inclusive, usar a solução fracionária para guiar o algoritmo!

Heurística Primal para Steiner

Heurística da Árvore Geradora Máxima:

- Seja x' a solução da relaxação linear atual
- Encontre uma árvore geradora máxima T de G usando x' como peso das arestas
- Enquanto houver folhas que não terminais, remova-as de T

A heurística prioriza arestas com maior valor na solução fracionária

Fixação de Variáveis por Implicações Lógicas

Seja $G' = (V', E')$ e Z' a instância obtida a partir de G e Z

- eliminado as arestas fixadas em 0
- contraindo as arestas fixadas em 1
  - Com o novo vértice sendo um terminal se a arestas era incidente a um terminal

Então:

- Caso exista $v \in Z'$ com grau 1 em G' então a aresta incidente a v pode ser fixada em 1
- Caso exista $v \in V \setminus Z$ com grau 1 em G', então a aresta correspondente pode ser fixada em 0
- Caso exista $v \in V \setminus Z$ com grau 2 em G', podemos fixar as arestas correspondentes em 0, adicionando uma aresta ligando os vizinhos de v em G' com custo igual à soma das arestas fixadas em zero
- Caso exista ponte de Steiner, a variável correspondente pode ser fixada em 1

Estratégias de Branching

Precisamos escolher uma variável para fazer o branching

- Normalmente escolhemos uma variável fracionária
- Mas até poderíamos escolher uma variável inteira...
  - Coloque $x_i \leq k$ e $x_i \geq k+1$ para k inteiro
- Uma ideia é escolher a variável "mais" fracionária
  - A mais próxima de $k + 1/2$ para qualquer inteiro k
- Outra ideia é escolher "menos" fracionária
  - A mais próxima de ser inteira
- Existem outras formas também que tentam pegar as variáveis mais relavantes de alguma forma

É possível fazer também branching nas inequações ao invés das variáveis

Por exemplo, escolhe um vértice $v \in V \setminus Z$ e crie dois subproblemas:

- No primeiro, coloque a equação $x(\delta(v)) = 0$
- No segundo, coloque a inequação $x(\delta(v)) \geq 1$

Percorrendo a Árvore

Precisamos pensar também como percorrer a árvore

Busca em Profundidade:

- Boa para encontrar novas soluções inteiras
- Mas pode ficar presa na subárvore muito tempo

Busca por Prioridade:

- Escolhemos de acordo com o valor da relaxação
- Por exemplo, aqueles com menor valor primeiro tem chances de ter soluções melhores
- Mas podemos começar a ficar com muitos nós ativos e gastar muita memória

Podemos também alternar durante a execução do algoritmo

Tamanho do Programa Linear

A resolução do PL é boa parte do tempo de execução do algoritmo

- Ele pode ser muito grande em termos de variáveis e restrições

Variáveis:

- Podemos começar algumas variáveis
- e adicionar outras apenas se elas podem entrar na base na iteração atual do Simplex

Restrições:

- Podemos remover restrições que não são satisfeitas com igualdade
- Guardando em um pool de restrições
- Quando formos gerar novas restrições, primeiro checamos o pool para encontrar restrições violadas
- Podemos descartar restrições muito antigas do pool e deixar elas serem geradas novamente