Programação Inteira

Matheus Souza D'Andrea Alves

2018.2

Sumário

Introdução

Infos gerais

Site

Salas:

- 2ª 306
- 4ª 202

Para a parte prática vamos implementar modelos e soluções usando o CPlex ## Programação linear

É um problema de otimização onde tanto a função objetivo quanto as restriçíes são lineares.

min/max c^tX

A__x eq 0

Modelagem

- Defina as variáveis do problema → como representar uma solução do problema
- Definir as restrições do problema → limites que definem o conjunto de pontos viáveis.
- A função objetivo → que vai ponderar cada solução

Fluxo máximo

Existe um grafo direcionado $G = (V, E)$ com um e apenas um vértice fonte e sumidouro, $\forall(i, j) \in E(G)$ tem uma capacidade $C_{i,j}$. Queremos maximizar a quantidade de produto que passa de $F \to S$

São minhas variáveis: $X_{i,j} \to \forall(i, j) \in E(G)$. Representando a quantidade de produto que sai de i e chega em j.

São minhas restrições: $X_{i,j} < C_{i,j}$;

É meu objetivo: $max\{\sum_{j \in N^+(S)} X_{j,s}\}$

Para solução

Métodos:

- Simplex (exponencial, rápido)
- Ponto Interiores (polinomial, rápido)

Modelagem

Categorização em formulação matemática

As categorias quando modelamos problemas em matemática caem nas seguintes categorias.

- Linear ou não linear
- Convexo ou não-convexo
- Contínuo ou Discreto
- Estocástica ou Determinismo

Dentro dessa categorização a programação inteira busca resolver os problemas:

- Discretos
- Determinísticos
- Não convexos
- Lineares e não lineares

Isso faz com que seja possível resolver a maioria dos problemas combinatórios propostos em computação.

Composição de um problema

Um problema de programação matemática é composto de :

- Variáveis de decisão
- Restrições
- Função objetivo
- Parametro de entrada

Um problema de programação inteira tem formato:

$$\begin{cases} min/max\{f(x)\} \\ g_i(x) \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b_i \\ x \in X | X \text{é discreto} \end{cases}$$

Definição:

Solução viável: valores atribuidos as variáveis que respeitam as Restrições. **Solução ótima**: é uma solução viável que maximiza/minimiza a função objetivo.

Forma padrão:

$$\begin{cases} maxC^T x \\ A_x \leq b \\ x \in Z_+^p * R_+^{n-p} \end{cases}$$

A melhor formulação possível, é uma formulação que defina a involtória convexa dos pontos inteiros, i.e. as o poliedro minimal que contém toda solução inteira, se isso acontecer conseguimos resolver através de PL, e logo resolver de forma polinomial.

Porém, a involtória convexa não é conhecida, ou sua representação é exponencial.

Definições

Considere duas formulações A e B para o mesmo PPI. Denominamos P_A e P_B seus poliedros equivalentes.

A formulação A é dita *tão forte quanto* a formulação B se $P_A \subseteq P_B$. Se a inclusão é estrita, isto é, $P_A \subset P_B$ então dizemos que A é uma formulação mais forte.

Se F é o conjunto de todas as viáveis soluções desse PPI, então temos que $Convo(F) \subseteq A$. Formulação ideal é aquela tal que $Convo(F) = P_A$

Modelando com variáveis inteiras.

Impor que uma ação x só pode ser feira se outra ação y for realizada.

$x, y \in [0, 1] | y \leq x$

Problema da Localização Facilitada:

- Conjunto de facilidades J
- Conjunto de Clientes I

Que facilidades precisam ser abertas para atender as demandas dos Clientes a um custo mínimo?

Seja $C_{i,j}$ o custo da facilidade j atender o cliente i. f_j o custo da abertura de uma facilidade.

Variáveis:

$$X_{j,i} = \begin{cases} 1, \text{ se } j \text{ atende } i \\ 0, \text{caso contrário.} \end{cases}$$

$$Y_j = \begin{cases} 1, \text{ se } j \text{ aberto} \\ 0, \text{caso contrário.} \end{cases}$$

Restrições:

$$X_{j,i} \in \{0,1\}, \forall j \in J, \forall i \in I$$

$$Y_j \in \{0, 1 \, \forall j \in J$$

$$\sum_{j \in J} X_{j,i}, \forall i \in I$$

$$|I| Y_j \geq \sum_{i \in I} X_{j,i}$$

Impor que uma variável X assumir apenas um dos valores de um conjunto finito $A = a_1, a_2, .., a_m$

$$Y_i = \begin{cases} 1, \text{se } X \text{ assume } a_i \\ 0, \text{caso contrário} \end{cases}.$$

$$\sum_{i=1}^{m} Y_i = 1$$

$$X = \sum_{i=1}^{m} Y_i a_i$$

Modelar funções lineares por partes

- Para cada intervalo que a slução x esteja temos uma função linear diferente
- A função f é conhecida apenas nos pontos a_i
- O valor de f é dado pela combinação linear de dois pontos consecutivos

  $\lambda f(a_i) + (1 - \lambda) f(a_{i+1})$

Variáveis:

$$Y_i = \begin{cases} 1, \text{se solução está no intervalo } [a_i, a_{i+1}] \\ 0, \text{caso contrário, } \forall i = \{\} \dots k \end{cases}.$$

$$\lambda_i \rightarrow \text{combinação linear}, \forall i = \{\} \dots k$$

Restrições:

$$min \sum_{i=1}^{k} \lambda_i = 1$$

$$min \sum_{i=1}^{k} Y_i = 1$$

$$\lambda_i \leq Y_i + Y_{i-1} | \forall i = 2, \dots, k$$

$$\lambda_i \leq Y_i$$

$$\lambda_i \geq 0, foralli = 1, \dots, k$$

$$Y_i \in \{0, 1\}, foralli = 1, \dots, k$$

Objetivo:

$$min \sum_{i=1}^{k} \lambda_i f(a_i)$$

Modelando restrições dijuntas

Suponha duas restrições:

- $a^T X \geq b*$
- $c^T X \geq d**$

Queremos que pelo menos uma delas sejam satisfeitas.

Variáveis:

$$Y = \begin{cases} 1, \text{se satisfaz } *. \\ 0, \text{se satisfaz } ** \end{cases}$$

Restrições:

$$a^t X \geq bY$$

$$c^t X \geq d(1 - Y)$$

Suponha que tenho agora k restrições: $a_i^t \geq b_i$, $\forall i \in [1..k]$, quero ativar p restrições.

minhas restrições extras são:

$$Y_i \in [0, 1], \forall i = 1..k$$

$$\sum_{i=1}^{k} Y_i = p$$

Exemplo do emparelhamento perfeito

Temos um grupo de n pessoas que precisam formar pares. Seja c_j o custo de parear pessoa i com j.

Queremos minimizar o custo dos emparelhamentos.

Variáveis:

$$x_{i,j} = \begin{cases} 1, \text{se satisfaz } i \text{ parear com } j. \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$\sum_{j \in E} x_{i,j} = 1$$

Objetivo:

$$min \sum_{j \in E} x_{i,j} c_{i,j}$$

Exemplo da coloração de vértices

Seja um Grafo G, definimos a coloração de G como a atribuição de uma entre k para cada vértice de forma que dada qqr aresta suas extremidades não compartilham cores.

Variáveis:

$$x_{i,j} = \begin{cases} 1, \text{se vértice } i \text{ é colorido com } j. \\ 0, \text{caso contrário} \end{cases}$$

$$w_j = \begin{cases} 1, \text{se } j \text{ for usado na coloração.} \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$\sum_{j=1}^{|V(G)|} X_{i,j} = 1$$

$$X_{i,j} + x_{k,j} \leq w_j, \forall i, k \in V(G)$$

Objetivo:

$$min \sum_{j=1}^{|V(G)|} w_j$$

Fortalecer modelo enfraquecendo variáveis

Exemplo: Lot Sizing

Seja $d_t \rightarrow$ demanda do tempo t; $f_t \rightarrow$ custo de produzir no tempo t; $p_t \rightarrow$ custo de produção por unidade; $h_t \rightarrow$ custo de armazenamento em t.

Modelagem padrão:

Variáveis:

$$X_t \rightarrow \text{Qtd produzida em } t$$

$$S_t \rightarrow \text{Qtd em estoque em } t$$

$$Y_t = \begin{cases} 1, \text{se algo foi produzido em } t. \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$S_m = 0$$
$$X_t \leq Y_t M$$
$$S_{t-1} + X_t = d_t + S_t$$

removendo M temos

$$X_t \leq Y_t \sum_{t=1}^{m} d_t$$

Objetivo:

$$min \sum_{t=1}^{m} (h_t S_t + f_t Y_t + p_t X_t)$$

Modelagem com fortalecimento

Variáveis:

$$W_{i,t} \rightarrow \text{Qtd produzida em } i \text{ para suprir a demanda em } t$$
$$S_t \rightarrow \text{Qtd em estoque em } t$$
$$Y_t = \begin{cases} 1, \text{se algo foi produzido em } t. \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$S_m = 0$$
$$S_1 = 0$$
$$W_{i,t} \leq Y_i d_t$$
$$S_{t-1} + \sum_{i=t}^{n} W_{t,i} = d_t + S_t$$

Objetivo:

$$min \sum_{t=1}^{m} (h_t S_t + f_t Y_t + p_t \sum_{i=t}^{n} W_{t,i})$$

Outra modelagem para coloração

Sabemos que coloração é equivalente a encontrar uma partição de G em k conjuntos idenpendentes maximais.

Dessa forma podemos escolher um vértice como *representante* de seu conjunto independente, nos levando as seguintes variáveis.

Variáveis:

$$X_{i,j} = \begin{cases} 1, \text{se o vértice } i \text{ e o vértice } j \text{ pertencem ao mesmo conjunto. Onde } i \leq j. \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$\sum_{N(v) \neq u \leq v} X_{u,v} = 1$$

$$X_{k,i} + X_{k,j} \leq X_{k,k}, \, \forall ij \in E, \forall k \in V, k \notin N[i] \sup N[j]$$

$$X_{i,j} \leq X_{i,i}$$

Caxeiro viajante

Variáveis

$$X_{i,j} = \begin{cases} 1, \text{se viajo de } i \text{ para } j. \text{ Onde } i \leq j. \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$\sum_{j \in N(i)} X_{i,j}$$

$$\sum_{j \in N(i)} X_{j,i}$$

$$\sum_{j \in V/S} \sum_{i \in S} X_{j,i} \geq 1$$

$$\sum_{j \in S} \sum_{i \in S} X_{j,i} \leq |S| - 1$$

Seja:

- J um conjunto de n tarefas
- M um conjunto de m máquinas
- Cada tarefa $j \in J$ temos a ordem de processamento $(\lambda_1^j, \lambda_2^j, ..., \lambda_m^j)$ para a execução de j.
- Para cada $j \in J$ e $i \in M$ temos o tempo de processamento $p_{i,j}$
- Uma tarefa é executada em cada máquina exatamente uma vez.

Seja minha entrada a definição da ordem das máquinas em que os jobs devem ser executados e o tempo em que cada λ_i^j leva para executar.

Variáveis

$$X_{j,t,i} = \begin{cases} 1, \text{se a tarefa } j\text{, começa no tempo } t\text{, na máquina } i. \\ 0, \text{caso contrário} \end{cases}$$

Problemas em árvores

Problema de minimizar o número de *branch*-vértices em uma árvore geradora.

> Dado um grafo $G = (V, E)$ encontrar uma árvore geradora que minimize o número de b-vértices.
>
> > Um vértice v é um b-vértice se na árvore se $d(v) \geq 3$

Variáveis:

$$X_e = \begin{cases} 1, \text{se a aresta } e \in E(G) \text{ pertence a árvore.} \\ 0, \text{caso contrário} \end{cases}$$

$$Y_v = \begin{cases} 1, \text{se o vértice } v \in V(G) \text{ é um } b\text{-vértice.} \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

mínimo

$$\sum_{e \in E(G)} X_e = n - 1$$

subciclo

$$\sum_{e \in E(S)} X_e = |S| - 1 \qquad \forall S \subset V$$

b-vértices

$$\sum_{e \in \delta(i)} X_e \leq Y_i \mathcal{M} + 2 \qquad \forall i \subset V$$

Objetivo:

$$min\{\sum_{v \in V} Y_v\}$$

Problema da k-tree mínima, encontrar uma árvore com k arestas de custo mínimo

$\omega_i \quad \forall i \in E(G) \rightarrow$ custo

Variáveis:

$$X_e = \begin{cases} 1, \text{se a aresta } e \in E(G) \text{ pertence a árvore.} \\ 0, \text{caso contrário} \end{cases}$$

$$Y_v = \begin{cases} 1, \text{se o vértice } v \in V(G) \text{ é um } b\text{-vértice.} \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$\sum_{e \in E(G)} X_e = k$$

$$\sum_{e \in \delta(i)} X_e \leq Y_i \mathcal{M}$$

$$\Updownarrow$$

$$X_i \leq Y_i \qquad , \forall i \in \delta(i)$$

$$\sum_{e \in E(S)} X_e \leq |S| - 1 \qquad , \forall S \subset V(G) \quad \text{and} \quad |S| \leq k + 1$$

$$\Updownarrow$$

$$\sum_{e \in E(S)} X_e \leq \sum_{i \in S\{j\}} Y_i \qquad , \forall S \subset V(G) \quad \text{and} \quad \forall j \in S$$

Objetivo

$$\sum_{e \in E} \omega_e X_e$$

Ring-star Problem

Dado um grafo misto $G = (V, E \cup A)$

$E = (u, v) \quad \forall u, v \in V(G) \quad \text{and} \quad u < v$

$A = \{u, v\} \quad u, v \in V(G)$

v_1 é denominado depósito

$c_{i,j}$ é o custo da aresta $(i, j) \in E$

$d_{i,j}$ é o custo da aresta $(i, j) \in A$

o ciclo é formado por arestas $e \in E$ e o ciclo(*Ring*) deve conter o depósito.

todo vértice que não faz parte do ciclo é conectado ao ciclo por um arco $a \in A$ de menor custo de conexão ao ciclo.

o custo do *Ring* é dado pela soma dos $c_{i,j}$ para i e j dentro do ciclo e $d_{i,j}$ para i ñ pertencente.

Variáveis:

$$X_e = \begin{cases} 1, \text{se a aresta } e \in E \text{ esta no ciclo.} \\ 0, \text{caso contrário} \end{cases}$$

$$Y_{i,j} = \begin{cases} 1, \text{se o vértice } i \text{ que não está no ciclo} \\ \quad \text{for ligado ao vértice } j \text{ pertencente ao ciclo.} \\ 0, \text{caso contrário} \end{cases}$$

Se $i \in V$ pertence ao cilo, $Y_{i,i} = 1$

Restrições:

$$\sum_{e \in \delta(i)} = 2Y_{i,i} \quad \forall i \in V$$

$$Y_{1,1} = 1$$

$$Y_{i,i} + \sum_{j \in N_a(i)} Y_{i,j} = 1$$

$$\sum_{i \in S\ j \notin S} X_{i,j} \geq 2Y_{k,k} \quad \forall S \subset V\{v_i\} \quad \forall k \in S$$

Coloração Equilibrada

A diferença na quantidade de vértices coloridos com cada cor não pode ser maior que 1

Variáveis:

$$X_{i,j} = \begin{cases} 1, \text{se a vértice } i \text{é colorido com } j. \\ 0, \text{caso contrário} \end{cases}$$

$$Y_j = \begin{cases} 1, \text{se a cor } j \text{ é usada} \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

$$\sum_{j=1}^{n} X_{i,j} = 1 \quad , \forall i \in V$$

$$X_{i,j} + X_{k,j} \leq 1 \quad , \forall i \in V \ , \forall k \in N(i) \ , \forall j = 1...n$$

Problema de alocação de Frequencia

Atribuição de frequencias à antenas de forma a minimizar interferência.

cada $(u, v) \in E(G)$ tem uma distância segura $d_{u,v} $

Existe um conjunto de frequencias F

Variáveis:

$$X_{i,j} = \begin{cases} 1, \text{se a antena } i \text{ é atribuida frequencia } j. \\ 0, \text{caso contrário} \end{cases}$$

Restrições:

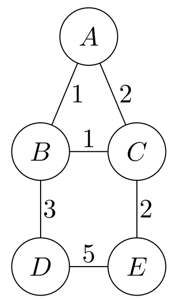

Figura 1:

$$\sum_{j \in F} X_{i,j} \leq \Delta(i) \quad ,\forall i \in V$$

$$X_{i,j} - X_{p,k} \leq 1 \quad \begin{cases} \forall i \in V \ , \forall p \in N(i) \\ \forall j \in F \ , \forall k \in F \end{cases}$$

Objetivo:

$$max \sum_{i \in V} \sum_{j \in F} X_i j$$

Máximo subgrafo balanceado

Dado um grafo $G = (V, E^+ \cup E^-)$, encontrar um grafo que pode ser particionado em no máximo k componentes equilibradas um grafo

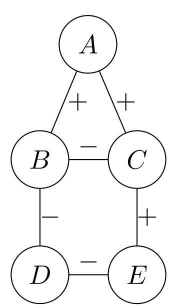

Figura 2:

Branch and bound

Definimos como branch and bound a estratégia de ramificar o problema em problemas mais limitados e resolver o todo por composição dos menores. Observe que tal estratégia leva a uma árvore.

Chamamos de nós ativos aqueles que não foram nem podados nem ramificados.

Os limitantes são classificados em:

- Limitantes inferiores (primais): heuristicas, etc...
- Limitantes superiores (duais): relaxação linear.

Na árvore de soluções existem três formas de percorrer em busca da melhor solução.

- Profundidade
  - foco em encontrar uma solução viável.
- Largura
  - foco na diversidade da busca.
- Limitantes
  - foco na qualidade do resultado.

Branching em um nó.

Uma variável é escolhida para ter seu valor limitado

Um nó podado não sofre mais *branch*, nosso objetivo é podar todos os nós da árvore.

Podamos das seguintes formas.

- Por otimalidade
  O nó 2 pode ser podado pois sabemos que a solução ótima de 2 é $\overline{Z} = underline Z = 20$
- Por limitante
  Melhor solução corrente: $\underline{Z} = 21$, podemos podar o nó 2 pois temos que $\underline{Z^2} < 21$
- Por inviabilidade
  Suponha o seguinte PPI: $8x_1 + 5x_2 + 3x_3 \leq 12$. observe que $x_1 = x_2 = 1 \implies 13 \leq 12$ logo podemos podar o nó 2 por inviabilidade.

Exemplo:

max $z = 4x_1 - x_2$

sujeito à

$7x_1 - 2x_2 \leq 14$

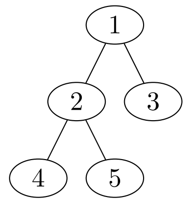

Figura 3:

$2x_1 - 2_x 2 \leq 3$
$x_2 \leq x_3$
$x_1, x_2 \in Z$

solução(1)
$x_1^* = 20/7, \quad x_2 = 3$
$\overline{z} = 59/7$
$\underline{z} = -\infty$

Ramificações de x_i:

x_i

$x_i \leq \lfloor x_i \rfloor \qquad x_i \geq \lceil x_i^* \rceil$
$x_1 \leq 2 \qquad x_1 \geq 3$
Podar nó direito.

solução(2)
$x_1^* = 2, \quad x_2^* = 1/2$
$\overline{z}^2 = 15/2$
$\underline{z} = -\infty$

Plano de corte

Branch and cut

Relaxação Lagrangiana

Geração de colunas em PI

Branch and Price

Teoria Poliédrica

Cortes

Faces

Facetas

abstract Function as base break case catch class checked continue default delegate do else enum event explicit extern false for foreach finally fixed goto if implicit in interface internal is lock namespace new not null operator out override params private protected public readonly ref return sealed sizeof stackalloc static struct switch this throw true try typeof unchecked unsafe using virtual while #if #else #elif #endif #define #undef #warning #error #line bool byte char const self decimal double float int long object uint ushort ulong sbyte short Set string void