

Geração de Colunas para Programação Inteira: Parte I

Eduardo Uchoa

Dep. Engenharia de Produção

Universidade Federal Fluminense

ELAVIO 2007

Conteúdo da Parte I

- Geração de colunas é basicamente uma técnica para resolver Programas Lineares (PLs) com muitas variáveis.
- Entretanto, essa técnica atualmente encontra grande importância na resolução de Programas Inteiros (PIs).

Conteúdo da Parte I

1. Algoritmo Simplex Revisado (PL)
2. Decomposição Dantzig-Wolfe (PL)
3. Problema de Bin-Packing / Cutting-Stock
(1ª aplicação de GC em um problema de PI)

Simplex Revisado (Dantzig, 1954)

- Método para resolver PLs de forma mais eficiente do que o Simplex tradicional por *tableaus* (Dantzig, 1947),
- Baseado na observação que o número de variáveis (n) geralmente é bem maior do que o número de restrições (m). (Aliás n sempre é $\geq m$)
- Notar que o simplex trabalha apenas com restrições de igualdade (=). Cada restrição de \leq ou \geq será convertida em = adicionando-se variáveis de folga ou de excesso.

Simplex Revisado

- Assuma que o PL está no seguinte formato:

$$\begin{array}{ll} \text{Max} & cx \\ \text{S.t.} & Ax = b \\ & x \geq 0 \end{array}$$

Exemplo

$$\begin{array}{llllll} \text{Max } z = & 19x_1 & +13x_2 & +12x_3 & +17x_4 & \\ \text{S.t.} & 3x_1 & +2x_2 & +1x_3 & +2x_4 & \leq 225 \\ & 1x_1 & +1x_2 & +1x_3 & +1x_4 & \leq 117 \\ & 4x_1 & +3x_2 & +3x_3 & +4x_4 & \leq 420 \\ & & & & & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

Exemplo

$$\text{Max } z = 19x_1 + 13x_2 + 12x_3 + 17x_4$$

$$\text{S.t.} \quad 3x_1 + 2x_2 + 1x_3 + 2x_4 + x_5 = 225$$

$$1x_1 + 1x_2 + 1x_3 + 1x_4 + x_6 = 117$$

$$4x_1 + 3x_2 + 3x_3 + 4x_4 + x_7 = 420$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0$$

Passo 1

Passo 1: Achar base inicial viável, ou seja, uma submatriz \mathbf{B} de \mathbf{A} tal que a solução do sistema $\mathbf{B}.\mathbf{x}_B = \mathbf{b}$ seja ≥ 0

No exemplo, digamos que $\{x_1, x_3, x_7\}$ formem a base

$$\mathbf{B} = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} \quad \mathbf{B} \cdot \begin{bmatrix} x_1 \\ x_3 \\ x_7 \end{bmatrix} = \begin{bmatrix} 225 \\ 117 \\ 420 \end{bmatrix} \Rightarrow \mathbf{x}_B = \begin{bmatrix} x_1 \\ x_3 \\ x_7 \end{bmatrix} = \begin{bmatrix} 54 \\ 63 \\ 15 \end{bmatrix} \quad z = 1782$$

Passo 2

Passo 2: Calcular variáveis duais π resolvendo o sistema $\pi \mathbf{B} = \mathbf{C}_B$.

No exemplo:

$$[\pi_1 \quad \pi_2 \quad \pi_3] \cdot \begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} = [19 \quad 12 \quad 0] \Rightarrow \pi = [3,5 \quad 8,5 \quad 0]$$

Passo 3

Passo 3: Calcular os custos reduzidos das variáveis não básicas x_j (*pricing*): $\bar{c}_j = c_j - \pi \mathbf{A}_j$

No exemplo:

$$\bar{c}_2 = 13 - [3,5 \quad 8,5 \quad 0] \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} \Rightarrow \bar{c}_2 = -2,5 \quad \bar{c}_4 = 17 - [3,5 \quad 8,5 \quad 0] \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix} \Rightarrow \bar{c}_4 = 1,5$$

$$\bar{c}_5 = 0 - [3,5 \quad 8,5 \quad 0] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \bar{c}_5 = -3,5 \quad \bar{c}_6 = 0 - [3,5 \quad 8,5 \quad 0] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow \bar{c}_6 = -8,5$$

Passo 3

Passo 3: Se nenhum custo reduzido for positivo, a solução é ótima. Senão uma variável com custo reduzido positivo deve ser escolhida para entrar na base.

No Exemplo, x_4 é a única opção,

Passo 4

Passo 4: Resolva o sistema $\mathbf{B} \cdot \mathbf{d} = \mathbf{A}_j$ onde \mathbf{A}_j é a coluna da variável j escolhida para entrar na base.

No Exemplo:

$$\begin{bmatrix} 3 & 1 & 0 \\ 1 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} d_1 \\ d_3 \\ d_7 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix} \Rightarrow \mathbf{d} = \begin{bmatrix} 0,5 \\ 0,5 \\ 0,5 \end{bmatrix}$$

Passo 5

Passo 5: Determine o maior t tal que $\mathbf{x}_B - t.\mathbf{d} \geq 0$

Se t for ilimitado o PL também é ilimitado.

No Exemplo:

$$\text{Max } t \text{ tal que } \begin{bmatrix} 54 \\ 63 \\ 15 \end{bmatrix} - t \begin{bmatrix} 0,5 \\ 0,5 \\ 0,5 \end{bmatrix} \geq 0 \quad \Rightarrow \quad t = 30$$

Passo 5

Passo 5: Uma das variáveis que limitaram o crescimento de t deve ser escolhida para sair da base.

No Exemplo, x_7 é a única opção.

Passo 6

Passo 6: Atualizar \mathbf{B} e \mathbf{x}_B . Depois voltar ao passo 2.

No exemplo, $\{x_1, x_3, x_4\}$ vão formar a nova base:

$$\mathbf{B} = \begin{bmatrix} 3 & 1 & 2 \\ 1 & 1 & 1 \\ 4 & 3 & 4 \end{bmatrix} \quad \mathbf{x}_B = \begin{bmatrix} 39 \\ 48 \\ 30 \end{bmatrix}$$

$$z = 1827$$

Vantagem do Simplex Revisado

- O único passo que depende do número de variáveis n é o *pricing* (Passo 3).
- Todos os demais passos são operações em matrizes $m \times m$.
- Grande vantagem em relação ao Simplex tradicional quando $n \gg m$.

Um Insight Fundamental

- É possível resolver PLs com um número gigantesco de variáveis (bilhões, trilhões, ...) desde que essas variáveis tenham uma estrutura especial que permita resolver o *pricing* de forma eficiente !
- Ou seja, ao invés de fazer os cálculos para cada variável individual, todo o passo do pricing pode ser resolvido como um problema de otimização !

Interpretação econômica do Simplex Revisado. Ex: Produção de Cadeiras

- Uma fábrica produz 50 cadeiras por semana, de um único modelo.
- A produção não pode ser aumentada porque o fornecimento de lâminas de madeira para revestimento e a quantidade de tecido são limitados.



50 lâminas/semana



75 metros/semana

Interpretação econômica: Produção de Cadeiras

- Para aumentar seu lucro, a empresa pretende diversificar sua produção. Para isso, precisa decidir quais modelos de cadeira deve produzir.



Interpretação econômica: Produção de Cadeiras



150

300

300

200



1

4

3

1



1

1

1

2



Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$



$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 \leq 50$$



$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 \leq 75$$



Interpretação econômica: Produção de Cadeiras

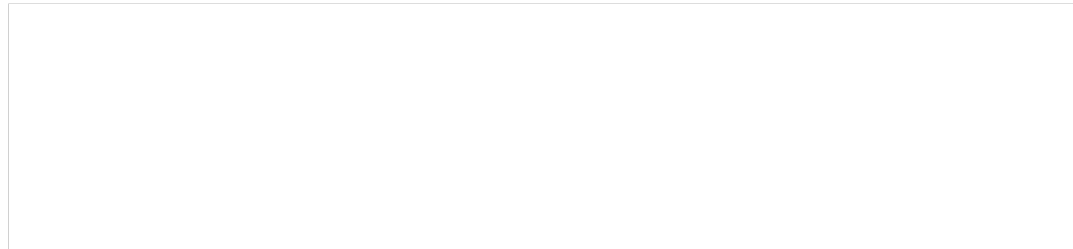


$$\max \quad 150 \quad x_1 \quad + \quad 300 \quad x_2 \quad + \quad 300 \quad x_3 \quad + \quad 200 \quad x_4$$

$$1 \quad x_1 \quad + \quad 4 \quad x_2 \quad + \quad 3 \quad x_3 \quad + \quad 1 \quad x_4 \quad - \quad x_5 \quad = \quad 50$$

$$1 \quad x_1 \quad + \quad 1 \quad x_2 \quad + \quad 1 \quad x_3 \quad + \quad 2 \quad x_4 \quad - \quad x_6 \quad = \quad 75$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$

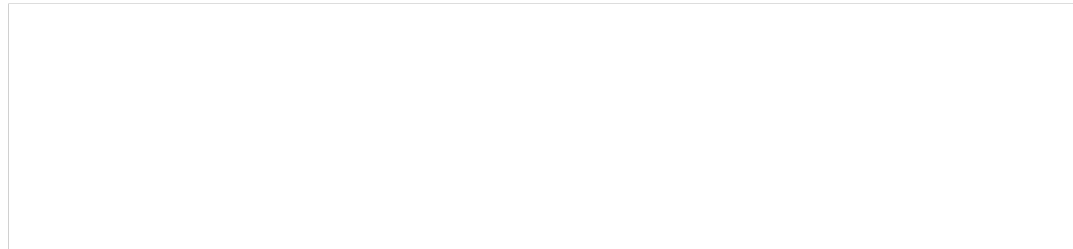
$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 - x_5 = 50 \quad \pi_1 = 150$$

$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 - x_6 = 75 \quad \pi_2 = 0$$

Produção atual = (50, 0, 0, 0, 0, 15)

Lucro atual = R\$ 7500,00

Interpretação econômica: Produção de Cadeiras

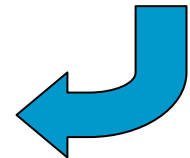


$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$

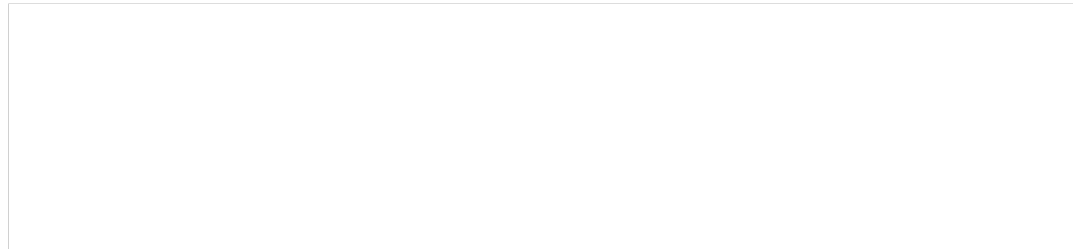
$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 - x_5 = 50 \quad \pi_1 = 150$$

$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 - x_6 = 75 \quad \pi_2 = 0$$

Considerando apenas a produção atual da empresa, cada lâmina de madeira a menos implica em uma redução nos lucros de R\$ 150,00



Interpretação econômica: Produção de Cadeiras

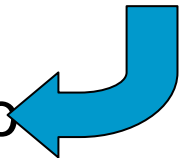


$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$

$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 - x_5 = 50 \quad \pi_1 = 150$$

$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 - x_6 = 75 \quad \pi_2 = 0$$

Considerando a produção atual, existe excesso de tecido disponível, uma redução em tecido (até certo ponto) não diminui o lucro.



Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$

$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 - x_5 = 50 \quad \pi_1 = 150$$

$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 - x_6 = 75 \quad \pi_2 = 0$$

$$\bar{c}_2 = ?$$

$$\bar{c}_3 = ?$$

$$\bar{c}_4 = ?$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 \quad x_1 \quad + \quad \mathbf{300} \quad x_2 \quad + \quad 300 \quad x_3 \quad + \quad 200 \quad x_4$$

$$1 \quad x_1 \quad + \quad \mathbf{4} \quad x_2 \quad + \quad 3 \quad x_3 \quad + \quad 1 \quad x_4 \quad - \quad x_5 \quad = \quad 50 \quad \pi_1 = \mathbf{150}$$

$$1 \quad x_1 \quad + \quad \mathbf{1} \quad x_2 \quad + \quad 1 \quad x_3 \quad + \quad 2 \quad x_4 \quad - \quad x_6 \quad = \quad 75 \quad \pi_2 = \mathbf{0}$$

$$\bar{c}_2 = ? 300$$

$$\bar{c}_3 = ?$$

$$\bar{c}_4 = ?$$

$$\bar{c}_2 = 300 - [\quad 150 \quad 0 \quad] \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$

$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 - x_5 = 50 \quad \pi_1 = 150$$

$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 - x_6 = 75 \quad \pi_2 = 0$$

$$\bar{c}_2 = -300$$

$$\bar{c}_3 = ?$$

$$\bar{c}_4 = ?$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 \quad x_1 \quad + \quad 300 \quad x_2 \quad + \quad \mathbf{300} \quad x_3 \quad + \quad 200 \quad x_4$$

$$1 \quad x_1 \quad + \quad 4 \quad x_2 \quad + \quad \mathbf{3} \quad x_3 \quad + \quad 1 \quad x_4 \quad - \quad x_5 \quad = \quad 50 \quad \pi_1 = \mathbf{150}$$

$$1 \quad x_1 \quad + \quad 1 \quad x_2 \quad + \quad \mathbf{1} \quad x_3 \quad + \quad 2 \quad x_4 \quad - \quad x_6 \quad = \quad 75 \quad \pi_2 = \mathbf{0}$$

$$\bar{c}_2 = -300$$

$$\bar{c}_3 = -150$$

$$\bar{c}_4 = ?$$

$$\bar{c}_3 = 300 - [\quad 150 \quad 0 \quad] \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 \quad x_1 \quad + \quad 300 \quad x_2 \quad + \quad 300 \quad x_3 \quad + \quad 200 \quad x_4$$

$$1 \quad x_1 \quad + \quad 4 \quad x_2 \quad + \quad 3 \quad x_3 \quad + \quad 1 \quad x_4 \quad - \quad x_5 \quad = \quad 50 \quad \pi_1 = 150$$

$$1 \quad x_1 \quad + \quad 1 \quad x_2 \quad + \quad 1 \quad x_3 \quad + \quad 2 \quad x_4 \quad - \quad x_6 \quad = \quad 75 \quad \pi_2 = 0$$

$$\bar{c}_2 = -300$$

$$\bar{c}_3 = -150$$

$$\bar{c}_4 = ?$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + \mathbf{200} x_4$$

$$1 x_1 + 4 x_2 + 3 x_3 + \mathbf{1} x_4 - x_5 = 50 \quad \pi_1 = \mathbf{150}$$

$$1 x_1 + 1 x_2 + 1 x_3 + \mathbf{2} x_4 - x_6 = 75 \quad \pi_2 = \mathbf{0}$$

$$\bar{c}_2 = -300$$

$$\bar{c}_3 = -150$$

$$\bar{c}_4 = 200$$

$$\bar{c}_4 = 200 - [150 \quad 0] \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 x_1 + 300 x_2 + 300 x_3 + 200 x_4$$

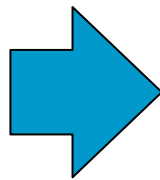
$$1 x_1 + 4 x_2 + 3 x_3 + 1 x_4 - x_5 = 50 \quad \pi_1 = 150$$

$$1 x_1 + 1 x_2 + 1 x_3 + 2 x_4 - x_6 = 75 \quad \pi_2 = 0$$

$$\bar{c}_2 = -300$$

$$\bar{c}_3 = -150$$

$$\bar{c}_4 = 200$$



Dos novos modelos, apenas
o modelo 4 pode aumentar
o lucro da empresa

Interpretação econômica: Produção de Cadeiras



$$\max \quad 150 \quad x_1 \quad + \quad 300 \quad x_2 \quad + \quad 300 \quad x_3 \quad + \quad 200 \quad x_4$$

$$1 \quad x_1 \quad + \quad 4 \quad x_2 \quad + \quad 3 \quad x_3 \quad + \quad 1 \quad x_4 \quad - \quad x_5 \quad = \quad 50 \quad \pi_1 = 100$$

$$1 \quad x_1 \quad + \quad 1 \quad x_2 \quad + \quad 1 \quad x_3 \quad + \quad 2 \quad x_4 \quad - \quad x_6 \quad = \quad 75 \quad \pi_2 = 50$$

Melhor opção = (25, 0, 0, 25, 0, 0)

Lucro = R\$ 8750,00

Decomposição Dantzig-Wolfe (1960)

- Um caso particular já tinha sido tratado por Ford e Fulkerson (1958).

Reformulação de um PL

- Seja (O) um PL cujas restrições podem ser divididas em dois conjuntos:

$$\begin{array}{llll} \text{Max} & cx & & \\ \text{S.t.} & Ax & = & b \\ & Dx & = & d \\ & x & \geq & 0 \end{array}$$

- Defina o poliedro $P = \{Dx = d, x \geq 0\}$

Reformulação de um PL

- O PL (O) equivale a
$$\begin{array}{ll}\text{Max} & cx \\ \text{S.t.} & Ax = b \\ & x \in P\end{array}$$
- Todo ponto contido em um poliedro pode ser descrito como uma combinação convexa de seus pontos extremos (+ seus raios extremos se ele for ilimitado, mas vamos supor que P é limitado)

Reformulação de um PL

- Ou seja todo ponto x em P pode ser escrito como uma combinação convexa de seus Q pontos extremos p_j .

$$x = \sum_{j=1}^Q p_j \lambda_j$$

$$\sum_{j=1}^Q \lambda_j = 1$$

$$\lambda \geq 0$$

Reformulação de um PL

- Substituindo x por seu equivalente

$$\begin{aligned}x &= \sum_{j=1}^Q p_j \lambda_j \\ \sum_{j=1}^Q \lambda_j &= 1 \\ \lambda &\geq 0\end{aligned}$$

$$\begin{array}{llll}\text{Max} & cx & & \\ \text{S.t.} & Ax & = & b \\ & x & \in & P\end{array}$$

\Downarrow

$$\begin{array}{llll}\text{Max} & \sum_{j=1}^Q (cp_j) \lambda_j & & \\ \text{S.t.} & \sum_{j=1}^Q (Ap_j) \lambda_j & = & b \\ & \sum_{j=1}^Q \lambda_j & = & 1 \\ & \lambda & \geq & 0\end{array}$$

Reformulação de um PL

- Temos um PL equivalente (M) com menos restrições (digamos $m+1$), mas com um número muito grande Q de variáveis.

$$\begin{array}{ll} \text{Max} & \sum_{j=1}^Q (cp_j) \lambda_j \\ \text{S.t.} & \sum_{j=1}^Q (Ap_j) \lambda_j = b \\ & \sum_{j=1}^Q \lambda_j = 1 \\ & \lambda \geq 0 \end{array}$$

Resolvendo pelo Simplex Revisado

- Começar resolvendo (M), o chamado PL *Mestre*, com apenas um pequeno subconjunto de suas variáveis suficiente para montar uma base.

$$\begin{array}{ll} \text{Max} & \sum_{j=1}^Q (cp_j) \lambda_j \\ \text{S.t.} & \sum_{j=1}^Q (Ap_j) \lambda_j = b \\ & \sum_{j=1}^Q \lambda_j = 1 \\ & \lambda \geq 0 \end{array}$$

Resolvendo pelo Simplex Revisado

- Sejam π (um vetor de dimensão m) e ν as variáveis duais associadas às restrições do Mestre

$$\begin{array}{ll} \text{Max} & \sum_{j=1}^Q (cp_j) \lambda_j \\ \text{S.t.} & \sum_{j=1}^Q (Ap_j) \lambda_j = b \quad (\pi) \\ & \sum_{j=1}^K \lambda_j = 1 \quad (\nu) \\ & \lambda \geq 0 \end{array}$$

Resolvendo pelo Simplex Revisado

- O passo do pricing pode ser resolvido de forma eficiente através do seguinte PL:

$$\begin{array}{llll} z^* = \text{Max} & (c - \pi A)x & - & v \\ \text{S.t.} & Dx & = & d \\ & x & \geq & 0 \end{array}$$

- Se z^* for positivo, a solução ótima x^* corresponde a um ponto extremo de P , cuja variável associada no mestre tem custo reduzido positivo e que pode entrar na base.

Resolvendo pelo Simplex Revisado

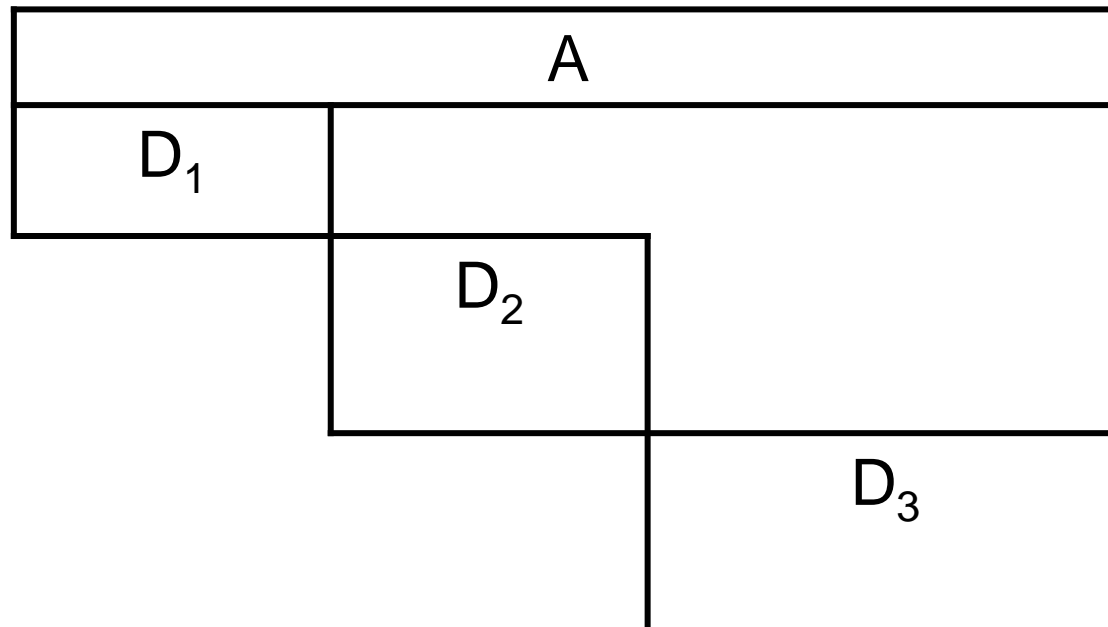
- Esse PL é define um *subproblema de pricing*.

$$\begin{array}{llll} z^* = \text{Max} & (c - \pi A)x & - & v \\ \text{S.t.} & Dx & = & d \\ & x & \geq & 0 \end{array}$$

- Se z^* for zero, fica provado que a solução corrente do Mestre é ótima.
- Normalmente, apenas uma fração muito pequena das Q variáveis chegará a entrar no Mestre.

Qual a vantagem da decomposição D-W para PL ?

- Alguns tipos de PLs podem ser decompostos de forma que o subproblema de pricing pode ser dividido em vários PLs pequenos independentes, que podem ser rapidamente resolvidos.



Qual a vantagem da decomposição D-W para PL ?

- Mas mesmo nesses casos a dec D-W ainda costuma ser mais lenta do que resolver (O) diretamente. Ou seja, raramente vale a pena usá-la para PL !
- A idéia da decomposição D-W acabou se revelando realmente útil em problemas de PI.

O problema do Bin Packing

- Dado um conjunto de n items, cada um com um peso w_i , colocá-los no menor número possível de caixas com capacidade C .
- Problema NP-difícil.
- Exemplo: $n=5$, $w_1=3$, $w_2=4$, $w_3=6$, $w_4=8$, $w_5=9$ e $C=10$.

O problema do Bin Packing

- Dado um conjunto de n items, cada um com um peso p_i , colocá-los no menor número possível de caixas com capacidade C .
- Problema NP-difícil.
- Exemplo: $n=5$, $w_1=3$, $w_2=4$, $w_3=5$, $w_4=8$, $w_5=9$ e $C=10$.

Nesse caso, são preciso 4 caixas.

O problema do Bin Packing

- O primeiro exemplo do uso de GC em PI foi nesse problema (Gilmore e Gomory, 1961, 1963).

Na verdade eles trataram de um problema ligeiramente mais genérico, o problema de cutting stock.

Formulação de Kantorovitch (1939,1960)

- Seja U um limite superior ao número de caixas necessárias.
- Defina variáveis y_j indicando se a caixa j vai ser usada.
- Defina variáveis x_{ij} indicando que o item i vai para a caixa j .

$$\begin{array}{ll} \text{Min} & \sum_{j=1}^U y_j \\ \text{S.t.} & \sum_{j=1}^U x_{ij} = 1 \quad i = 1 \dots n \\ & \sum_{i=1}^n w_i x_{ij} \leq C \cdot y_j \quad j = 1 \dots U \\ & x, y \in \{0,1\}^{n(U+1)} \end{array}$$

Formulação de Kantorovitch

- Essa formulação não funciona na prática
 - O limite inferior da sua relaxação linear é ruim, igual ao limite trivial $\sum_{i=1}^n w_i / C$.
 - No exemplo, esse limite seria 2,9.
 - A simetria das variáveis faz com que algoritmos para PI, como o branch-and-bound, sejam ineficientes.

Formulação de Gilmore-Gomory

- Defina uma variável para cada uma das Q possíveis combinações de itens em caixas. O número Q pode ser muito grande !
- Com $n=5$, $w_1=3$, $w_2=4$, $w_3=5$, $w_4=8$, $w_5=9$ e $w=10$; são 8 combinações: $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{5\}$, $\{1,2\}$, $\{1,3\}$ $\{2,3\}$.

Formulação de Gilmore-Gomory

- Defina o coeficiente a_{ij} como sendo 1 se o item i está no padrão j e 0 caso contrário.

$$\begin{array}{ll} \text{Min} & \sum_{j=1}^Q \lambda_j \\ \text{S.t.} & \sum_{j=1}^Q a_{ij} \lambda_j = 1 \quad i = 1 \dots n \\ & \lambda \in \{0,1\}^Q \end{array}$$

Formulação de Gilmore-Gomory

■ No exemplo:

$$\begin{array}{llllllll}
 \text{Min} & \lambda_1 & + \lambda_2 & + \lambda_3 & + \lambda_4 & + \lambda_5 & + \lambda_6 & + \lambda_7 & + \lambda_8 & & \\
 \text{S.t.} & \lambda_1 & & & & & + \lambda_6 & + \lambda_7 & & = & 1 \\
 & & \lambda_2 & & & & + \lambda_6 & & + \lambda_8 & = & 1 \\
 & & & \lambda_3 & & & & + \lambda_7 & + \lambda_8 & = & 1 \\
 & & & & \lambda_4 & & & & & = & 1 \\
 & & & & & \lambda_5 & & & & = & 1 \\
 & & & & & & & & \lambda & \in & \{0,1\}^8
 \end{array}$$

Formulação de Gilmore-Gomory

- No exemplo acima, o limite obtido pela relaxação linear da formulação é 3,5.
- Os limites obtidos pelo G-G são extremamente fortes.
 - Raramente se encontra uma instância em que esse limite arredondado para cima não iguale o valor da solução ótima.
 - Nunca se achou uma instância em que esse limite arredondado para cima estivesse a mais de 1 unidade da solução ótima !

Resolvendo por GC

$$\begin{array}{ll} \text{Min} & \sum_{j=1}^Q \lambda_j \\ \text{S.t.} & \sum_{j=1}^Q a_{ij} \lambda_j = 1 \quad i = 1 \dots n \\ & \lambda \geq 0 \end{array}$$

Seja π o vetor de variáveis duais associadas as m restrições

Subproblema de Pricing

$$\begin{array}{ll} \text{Min} & 1 - \sum_{i=1}^n \pi_i x_i \\ \text{S.t.} & \sum_{i=1}^n w_i x_i \leq C \\ & x \in \{0,1\}^n \end{array}$$

Esse é um clássico problema da mochila (knapsack), que é NP-difícil, mas muito bem resolvido na prática.

De fato, existe algoritmo pseudo-polinomial ($O(mC)$) para esse problema.

Formulação de Gilmore-Gomory

- Gilmore e Gomory (1961, 1963) apenas resolviam a relaxação linear, a solução inteira era encontrada heurísticamente por arredondamento.
- A abordagem funcionava muito bem na prática, as soluções assim obtidas tinham garantias de estarem muito próximas do ótimo.
- Entretanto, o uso dessa GC em algoritmos exatos (que sempre encontram a solução ótima) só aconteceu após os anos 80, quando se criou o método de *branch-and-price*.

Geração de Colunas para Programação Inteira: Parte II

Eduardo Uchoa

Dep. Engenharia de Produção

Universidade Federal Fluminense

ELAVIO 2007

Conteúdo da Parte II

- Nos anos 80 a geração de colunas foi combinada com a técnica de branch-and-bound para obter um novo tipo de algoritmo exato para problemas de PI.
- Os chamados algoritmos de *branch-and-price* obtiveram sucesso em vários problemas importantes de PI.

Conteúdo da Parte II

1. Decomposição D-W para PI
2. Algoritmo de *branch-and-price*
3. Casos de sucesso

Programação Inteira

- Resolver problemas no seguinte formato:

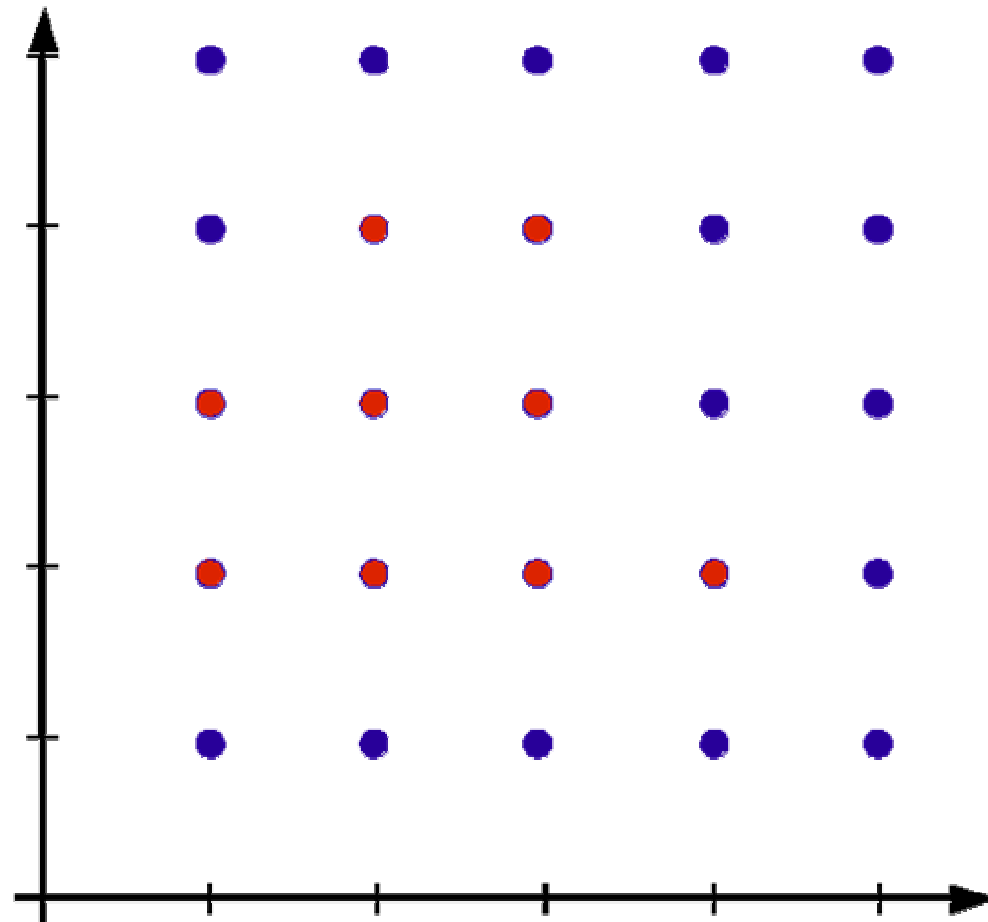
$$\begin{array}{ll} \text{Max} & cx \\ \text{S.t.} & Ax = b \\ & x \in Z_+^n \end{array}$$

- Ou seja, achar o melhor ponto inteiro satisfazendo um conjunto de restrições lineares

Programação Inteira

- Normalmente um PI surge como uma formulação de um Problema de Otimização Combinatória (POC).
- A idéia é representar cada possível solução do POC como um ponto num certo espaço de variáveis. Seja X o conjunto desse pontos.

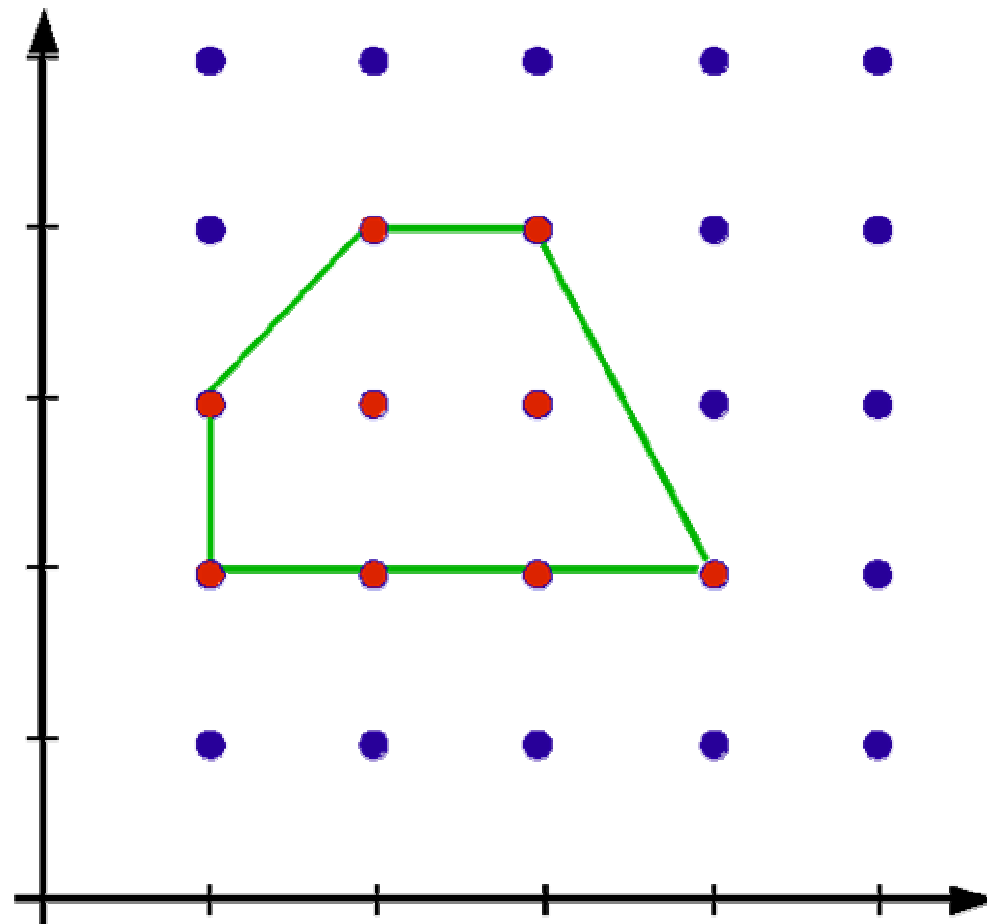
Conjunto de soluções viáveis



Programação Inteira

- Idealmente, caso se conhecesse as restrições que definem a envoltória convexa desses pontos ($\text{Conv}(X)$), o POC poderia ser resolvido como um simples PL.

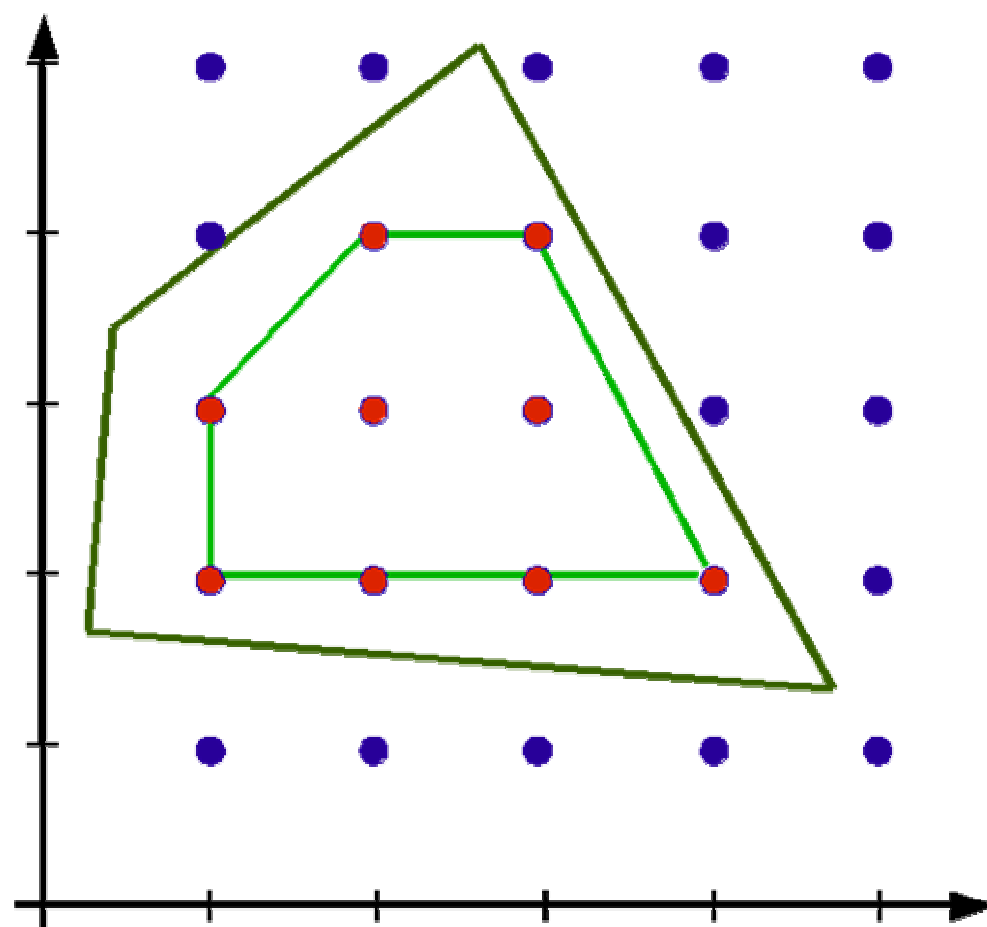
Envoltória Convexa



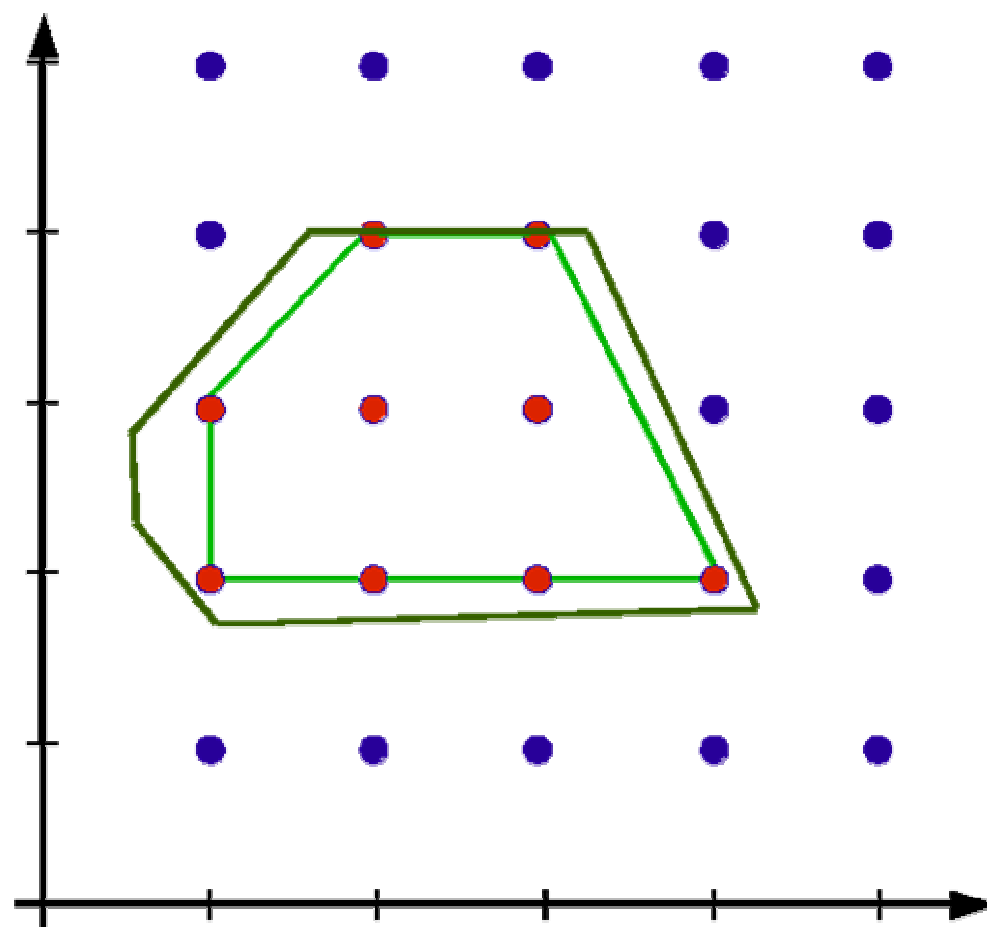
Programação Inteira

- Entretanto, se o POC for NP-difícil, a menos que $P=NP$, nunca será possível obter todas as restrições de $\text{Conv}(X)$.
- O que se pode fazer é obter *formulações*, ou seja, poliedros contendo todos os pontos de X , mas nenhum ponto inteiro fora de X .
- O mesmo problema admite inúmeras formulações. Quando mais próxima a formulação estiver de $\text{Conv}(X)$, melhor será o desempenho dos algoritmos de PI usados para resolver o POC.
- Em particular, quanto melhor a formulação, mais o valor obtido ao se resolver a relaxação linear do PI estará próximo do valor ótimo.

Formulação 1



Formulação 2



Decomposição para PI

- Seja (O) um PI cujas restrições podem ser divididas em dois conjuntos:

$$\begin{array}{ll} \text{Max} & cx \\ \text{S.t.} & Ax = b \\ & Dx = d \\ & x \in Z_+^n \end{array}$$

- Defina o conjunto $P = \{Dx = d, x \in Z_+^n\}$

Reformulação de um PI

- O PI (O) equivale a
$$\begin{array}{ll} \text{Max} & cx \\ \text{S.t.} & Ax = b \\ & x \in P \end{array}$$
- Suponha que P contenha um número finito Q de pontos inteiros p_1, \dots, p_Q .

Reformulação de um PI

- Todo ponto em P pode ser escrito trivialmente como uma combinação convexa inteira de seus Q pontos.

$$x = \sum_{j=1}^Q p_j \lambda_j$$

$$\sum_{j=1}^Q \lambda_j = 1$$

$$\lambda \in \{0,1\}^Q$$

Reformulação de um PI

- Substituindo x por seu equivalente

$$x = \sum_{j=1}^Q p_j \lambda_j$$

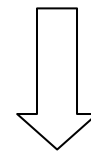
$$\sum_{j=1}^Q \lambda_j = 1$$

$$\lambda \in \{0,1\}^Q$$

$$\text{Max} \quad cx$$

$$\text{S.t.} \quad Ax = b$$

$$x \in P$$



$$\text{Max} \quad \sum_{j=1}^Q (cp_j) \lambda_j$$

$$\text{S.t.} \quad \sum_{j=1}^Q (Ap_j) \lambda_j = b$$

$$\sum_{j=1}^Q \lambda_j = 1$$

$$\lambda \in \{0,1\}^Q$$

Relaxação Linear do PI reformulado

- Esse PL pode ser resolvido por GC.
- Mas nesse caso, o valor da relaxação linear do PI reformulado pode ser melhor do que a relaxação linear do PI original (O) !

$$\begin{array}{ll} \text{Max} & \sum_{j=1}^Q (cp_j) \lambda_j \\ \text{S.t.} & \sum_{j=1}^Q (Ap_j) \lambda_j = b \\ & \sum_{j=1}^Q \lambda_j = 1 \\ & \lambda \geq 0 \end{array}$$

Relaxação Linear do PI reformulado

- Isso acontece porque a integralidade não está sendo relaxada no subproblema.

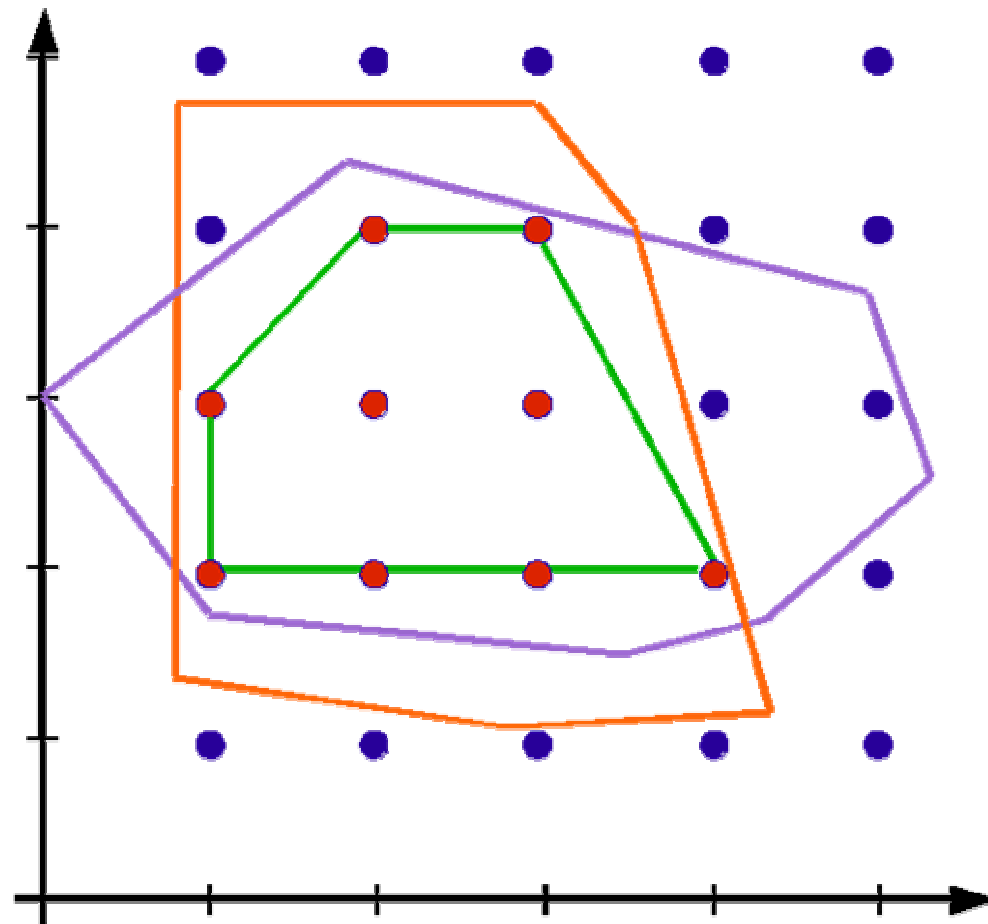
$$\begin{array}{llll} z^* = \text{Max} & (c - \pi A)x & - & v \\ \text{S.t.} & Dx & = & d \\ & x & \in & Z_+^n \end{array}$$

Relaxação Linear do PI reformulado

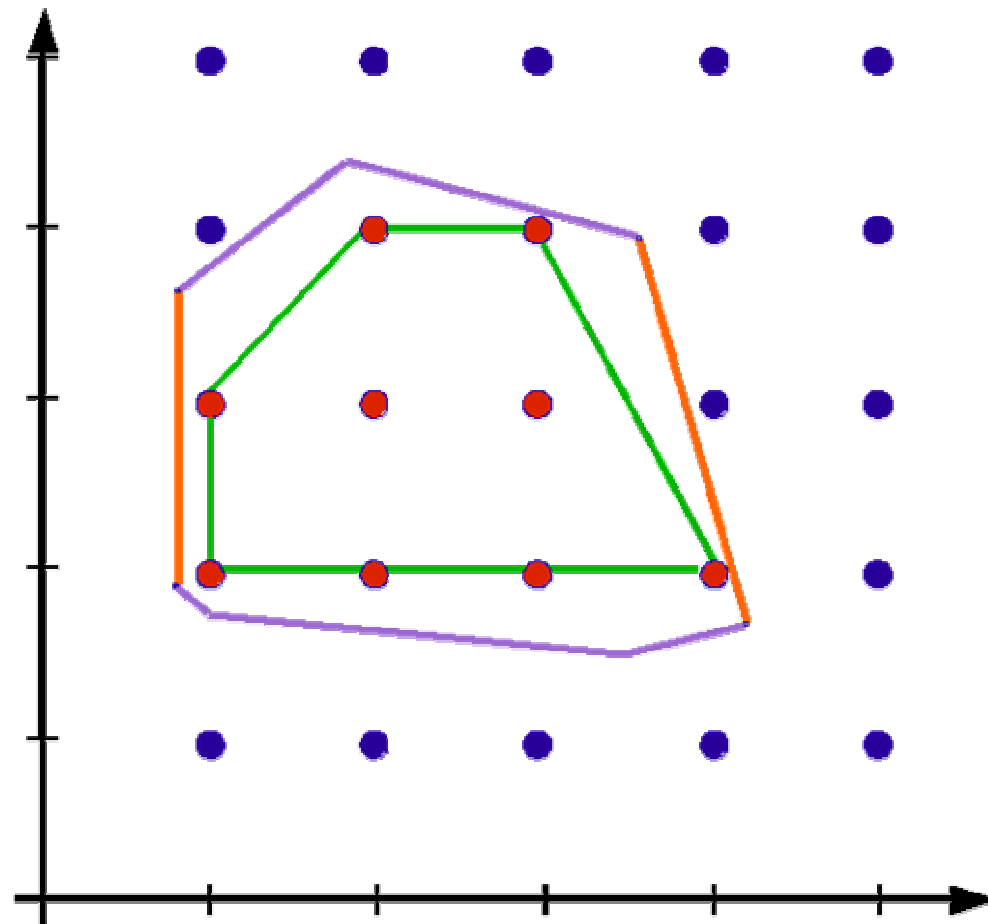
$$\begin{array}{ll}
 \text{Max} & \sum_{j=1}^Q (cp_j) \lambda_j \\
 \text{S.t.} & \sum_{j=1}^Q (Ap_j) \lambda_j = b \\
 & \sum_{j=1}^Q \lambda_j = 1 \\
 & \lambda \geq 0
 \end{array}
 \quad \longleftrightarrow \quad
 \begin{array}{ll}
 \text{Max} & cx \\
 \text{S.t.} & Ax = b \\
 & x \in \text{Conv}\{Dx = d, d \in Z_+^n\} \\
 & x \geq 0
 \end{array}$$

A reformulação equivale a convexificar as restrições $Dx=d$ do PI. Notar que se o subproblema for NP-difícil, uma descrição explícita desse poliedro provavelmente nunca será conhecida.

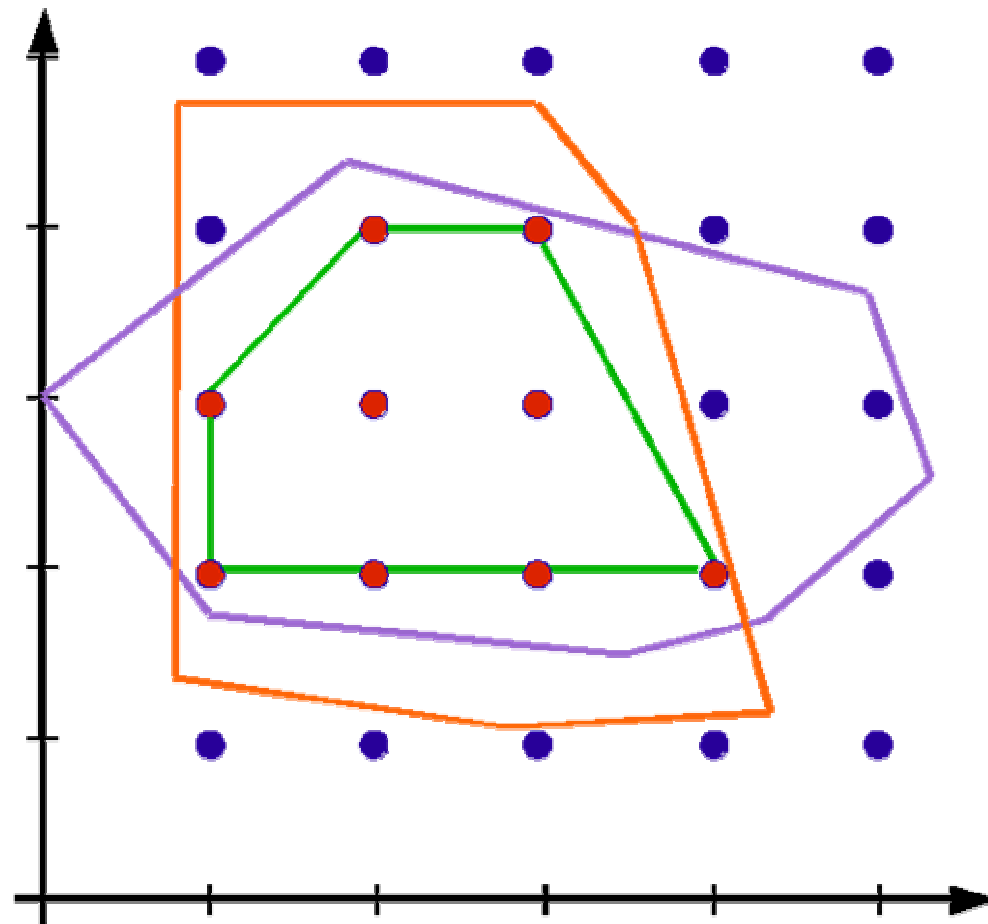
Uma formulação dividida em dois conjuntos de restrições



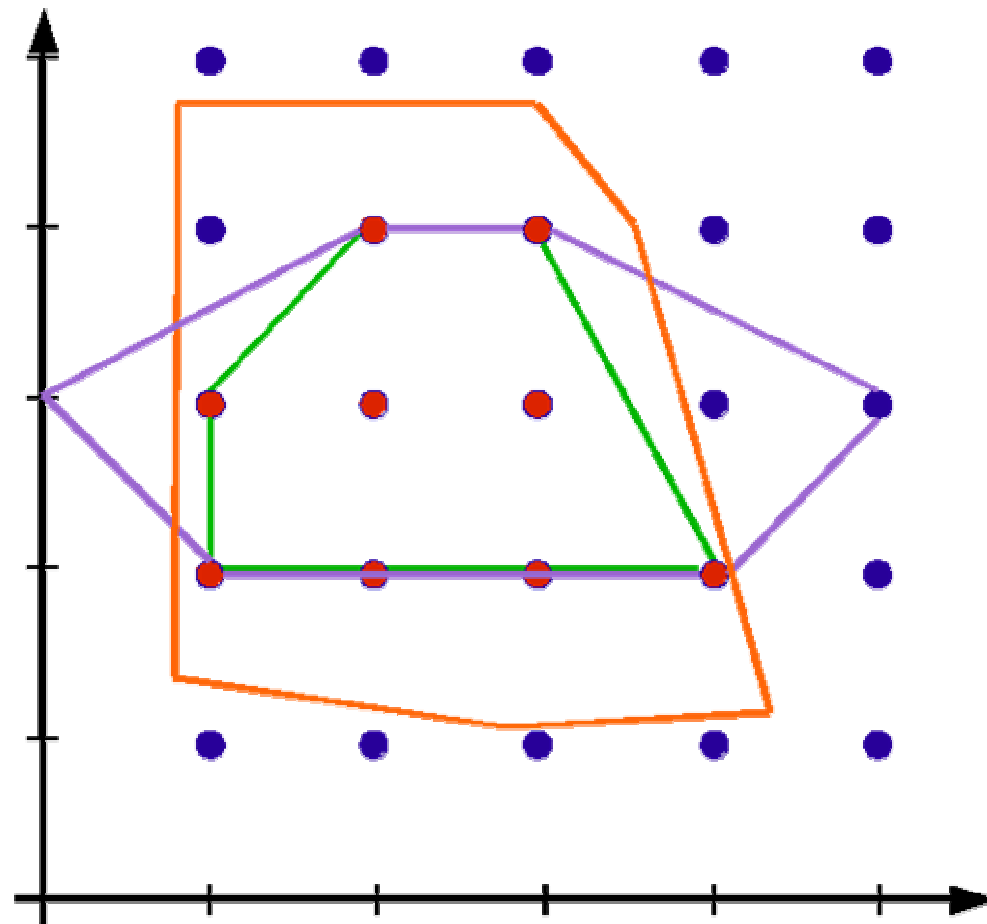
A formulação é a intersecção dos
dois poliedros



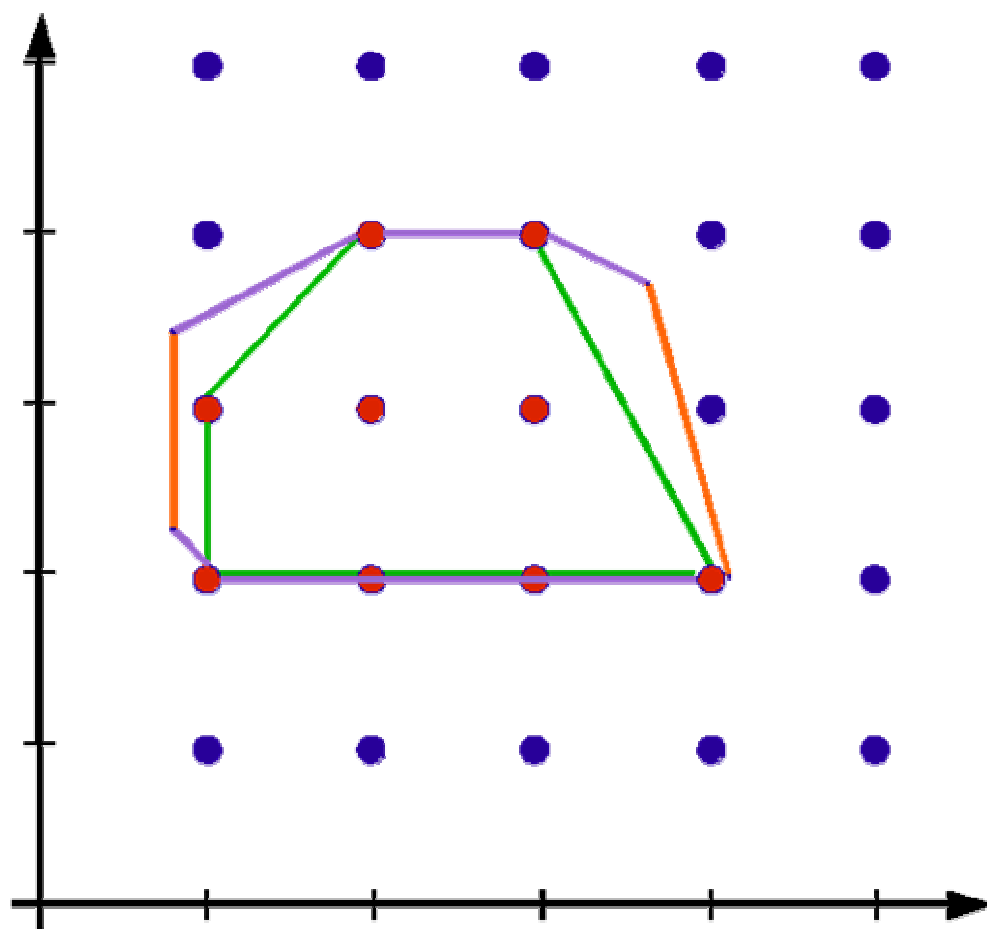
Convexificando um dos conjuntos de restrições



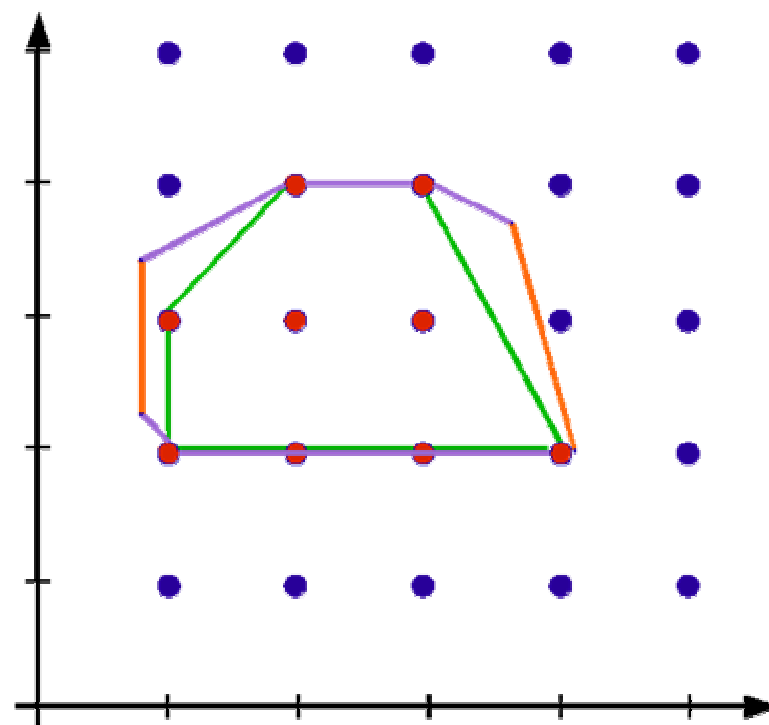
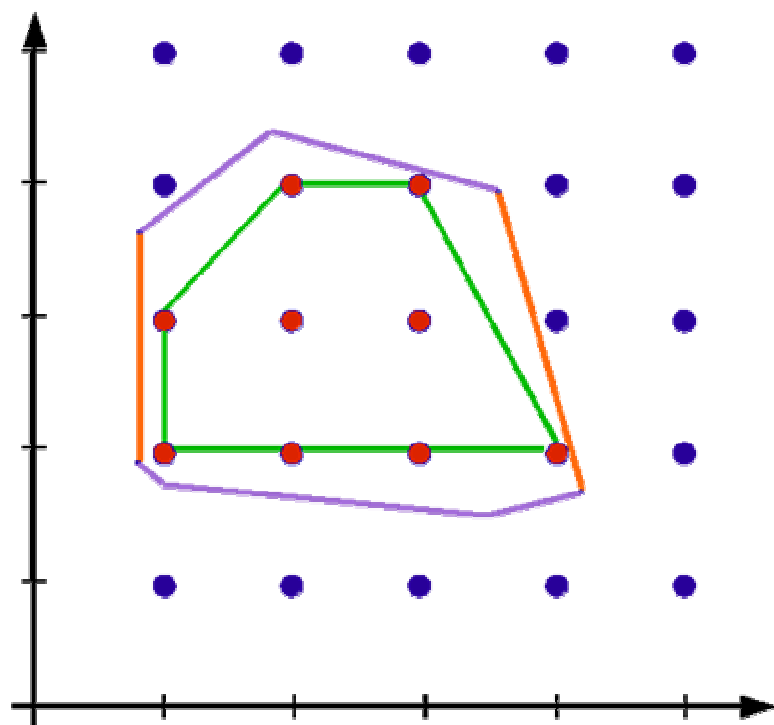
Convexificando um dos conjuntos de restrições



Nova formulação fortalecida



Comparação



Exemplo do Bin Packing

- A formulação forte de Gilmore-Gomory pode ser obtida diretamente através da decomposição para PI da formulação fraca de Kantorovitch.
- O fortalecimento decorre diretamente da convexificação das restrições de mochila.

Equívoco encontrado na literatura

Alguns autores afirmam que deve-se mandar as restrições “fáceis” para o subproblema e manter as “complicadoras” no mestre. Assim o pricing inteiro vai poder ser bem resolvido.

- Por exemplo, restrições que levem a subproblemas resolvidos como árvores geradoras, fluxos em rede, matchings, etc.

Realidade: No pain, no gain.

- Em geração de colunas para PI o *pricing* deve ser NP-difícil ! É isso que faz com que a convexificação parcial gere grandes melhorias.
- Entretanto o subproblema realmente deve ser bem tratável na prática. Boas opções:
 - Problemas resolvidos por algoritmos pseudo-polinomiais. Ex: mochila ($O(nC)$)
 - Problemas que continuem pseudo-polinomiais para algum parâmetro de controle fixo. Ex: caminho mais curto com capacidade e eliminação de s-ciclo ($O(s!n^2C)$)

Realidade: No pain, no gain.

- Uma observação similar vale para a escolha do subproblema quando se usa relaxação lagrangeana em problemas de PI.

Branch-and-Price

- Somente nos anos 80 colocou-se a idéia de aproveitar as formulações fortalecidas pela decomposição inteira para criar algoritmos exatos.
- A GC foi combinada com a mais tradicional técnica para PI, o branch-and-bound (Land e Doig, 1960), para criar os algoritmos de branch-and-price.
- Essa combinação oferece algumas dificuldades a serem superadas.

Como fazer o branching ?

- A idéia mais imediata é fazer branching sobre variáveis lambda do mestre que estejam fracionárias ao final da GC.
 - Por exemplo, em um subproblema λ_j pode ser obrigada a ser ≥ 1 e no outro a ser 0.
- Fixar uma variável a 0 significa retirar essa variável da base e proibi-la de entrar de volta. O problema é que essa variável pode ser a solução ótima do subproblema de pricing. Nesse caso, o pricing tem que ser alterado para retornar a segunda melhor solução do subproblema !

Como fazer o branching ?

- Outra opção é fazer branching sobre uma variável x da formulação original (uma solução fracionária sobre λ pode ser convertida em uma solução fracionária sobre x).
 - Isso em geral não provoca alterações no subproblema.
- O problema é que muitas vezes (como no caso do bin packing) as variáveis do problema original tem simetria. O branching se torna inefetivo.

Como fazer o branching ?

- A verdade é que ainda não existe um consenso sobre uma boa regra de branching geral para branch-and-price !
- Dezenas de autores vem discutindo a questão nos últimos 20 anos.

O problema da convergência

- Toda GC está sujeita a problemas de convergência, muitas iterações até que o subproblema prove que a solução corrente do Mestre é ótima.
- Como um branch-and-price pode necessitar explorar milhares de nós na sua árvore de enumeração, o problema pode se tornar particularmente sério.

O problema da convergência

- Dezenas de propostas existem para acelerar a convergência de uma GC, especialmente no contexto de um branch-and-price.
- Técnicas que tem tido particular sucesso nos últimos anos partiram da idéia de *estabilização dual* (Du Merle et al. 1999).
 - Basicamente isso consiste em obter boas estimativas para os valores ótimos das variáveis duais para fazer com que o subproblema gere logo “boas colunas”.

Casos de Sucesso

- Apesar das dificuldades apontadas anteriormente, os algoritmos de branch-and-price *funcionam bem* para algumas classes importantes de problemas:
 - Roteamento de veículos com janelas de tempo (1º BP bem sucedido, Desrosiers, Soumis e Desrochers, 1984).
 - Crew Scheduling
 - Max SAT
 - Multi-fluxo inteiro

Casos de Sucesso

- Bin packing e Cutting Stock
- Coloração de grafos
- Generalized Assignment
- Clustering
- ...

Branch-and-Price para o Problema de Alocação Generalizada (Dissertação de Mestrado, 2003)

Alexandre Altoé Pigatti

Orientadores :

Marcus Vinicius Soledade Poggi de Aragão

Eduardo Uchoa

Problema de Alocação Generalizada (PAG)

- Consiste em alocar um conjunto de tarefas $J=\{1,2,\dots,n\}$ a um conjunto de agentes $I=\{1,2,\dots,m\}$
- Cada agente i possui uma quantidade limitada de um determinado recurso.
- Alocar a tarefa j ao agente i acarreta um custo c_{ij} e consome do agente i uma quantidade a_{ij} de recurso.
- O objetivo é encontrar a alocação de tarefas aos agentes de custo mínimo.
- É um problema NP-difícil

Exemplo PAG

	<i>j1</i>	<i>j2</i>	<i>j3</i>	<i>j4</i>	<i>j5</i>	<i>j6</i>	<i>j7</i>	<i>j8</i>	S
<i>i1</i>	10	15	25	4	5	8	12	5	
<i>i2</i>	11	52	8	16	9	7	15	8	
<i>i3</i>	52	13	8	54	7	6	14	9	

Exemplo PAG resolvido

	<i>j1</i>	<i>j2</i>	<i>j3</i>	<i>j4</i>	<i>j5</i>	<i>j6</i>	<i>j7</i>	<i>j8</i>	<i>S</i>
<i>i1</i>	10	15	25	4	5	8	12	5	5 4 15
<i>i2</i>	11	52	8	16	9	7	15	8	8 11
<i>i3</i>	52	13	8	54	7	6	14	9	9 14 6

Formulação clássica do PAG

I : conjunto de agentes ($i = 1, 2, \dots, m$).

J : conjunto de tarefas ($j = 1, 2, \dots, n$).

b_i = capacidade do agente i

a_{ij} = recurso consumido pela tarefa j quando ela é alocada ao agente i

c_{ij} = custo de se alocar a tarefa j ao agente i

$$x_{ij} = \begin{cases} 1; \text{ tarefa } j \text{ é alocada ao agente } i \\ 0; \text{ caso contrário} \end{cases}$$

$$\text{PAG-C} \left\{ \begin{array}{l} (1) \quad \min f(x) = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \\ \text{sujeito a} \\ (2) \quad \sum_{j=1}^n a_{ij} x_{ij} \leq b_i; i = 1, 2, \dots, m \\ (3) \quad \sum_{i=1}^m x_{ij} = 1; j = 1, 2, \dots, n \\ (4) \quad x_{ij} \in \{0, 1\} i = 1, 2, \dots, m; j = 1, 2, \dots, n \end{array} \right.$$

Formulação do PAG por número exponencial de variáveis

- $K_i = \{v_i^1, v_i^2, \dots, v_i^{k_i}\}$ o conjunto de todas as alocações possíveis de tarefas para o agente i .

- $v_i^k = \{v_{i1}^k, v_{i2}^k, \dots, v_{in}^k\}$ uma solução viável para:

$$\sum_{j=1}^n a_{ij} v_{ij}^k \leq b_i$$

$$v_{ij}^k \in \{0, 1\} \quad j \in \{1, \dots, n\},$$

- Seja y_i^k para $i \in \{1, \dots, m\}$ e $k \in K_i$ uma variável binária indicando se uma alocação viável v_i^k é selecionada para o agente i

$$\text{PAG-Exp} \left\{ \begin{array}{l} (1) \quad \min \sum_{i=1}^m \sum_{k=1}^{k_i} \left(\sum_{j=1}^n c_{ij} v_{ij}^k \right) y_i^k \\ \text{sujeito a} \\ (2) \quad \sum_{i=1}^m \sum_{k=1}^{k_i} v_{ij}^k y_i^k = 1 \quad j \in \{1, \dots, n\}, \\ (3) \quad \sum_{k=1}^{k_i} y_i^k \leq 1 \quad i \in \{1, \dots, m\}, \\ (4) \quad y_i^k \in \{0, 1\} \quad i \in \{1, \dots, m\}, k \in K_i \end{array} \right.$$

Formulação do PAG por número exponencial de variáveis

O problema da mochila inteira associada ao agente i na formulação clássica (PAG-C) é dado como:

$$\begin{aligned} & \min \sum_{j=1}^n c_{ij} x_{ij} \\ & \text{s.a.} \\ & \sum_{j=1}^n a_{ij} x_{ij} \leq b_i, \\ & x_{ij} \in \{0, 1\} \quad j \in \{1, \dots, n\} \end{aligned}$$

Na (PAG-Exp) as restrições de capacidades dos agentes são representadas pela restrição:

$$\sum_{k=1}^{k_i} y_i^k \leq 1$$

Formulação do PAG por número exponencial de variáveis

isto permite que a solução ótima para a i -ésima mochila (agente do PAG) , seja obtida ao se otimizar segundo a função objetivo:

$$\min \sum_{k=1}^{k_i} \left(\sum_{j=1}^n c_{ij} v_{ij}^k \right) y_i^k$$

Resolução exata do PAG

- Geração de Colunas
- Estabilização da Geração de Colunas
- Branch-and-Price

Geração de Colunas

- Permite resolver problemas com número exponencial de variáveis
- Inicialização: artificial, heurística
- Sub-problema de geração de colunas: Avaliação implícita dos custos reduzidos de todas as variáveis.
- Iteração: Resolve-se o LP. Resolve-se o Subproblema usando os valores das variáveis duais de forma que a sua solução indica as colunas que teriam o custo reduzido mais negativo se fossem adicionadas ao LP.
- Otimalidade: Quando não existir nenhuma coluna de custo reduzido negativo a solução do LP restrito é equivalente a uma solução do LP completo.

Algoritmo Geração de Colunas

```
conjCol = mestreReduzidoInicial();
aumenta = true;
while aumenta do
    valorSolucao = resolve(conjCol);
    if numColunas(conjCol) > numMaxCols then
        | retiraColunas(conjCol, range);
        | resolve(conjCol);
    aumenta = false;
    colunas = 0;
    for i = 1; i < m; i++ do
        | colunas+ = resolveSubProblema(i, &gerouColuna);
        | if gerouColuna then
            | | aumenta = true;
    if aumenta then
        | conjCol+ = colunas;
return valorSolucao;
```


Subproblema Geração de Colunas

$$z(K P_i) = \max \sum_{1 \leq j \leq n} (u_j - c_{ij}) x_{ij}$$

s.d.

$$\sum_{1 \leq j \leq n} a_{ij} x_j^i \leq b_i$$

$$x_j^i \in \{0, 1\} \quad j \in \{1, \dots, n\}$$

Onde u_j é valor da variável dual relativa a restrição de obrigatoriedade de alocação da tarefa j .

Subproblema Geração de Colunas

$$\min_{1 \leq i \leq m} \{z(KP_i) - v_i\}$$

Para se calcular o valor do custo reduzido da coluna ainda é preciso levar em consideração o valor da variável dual v_i associada a restrição de capacidade do agente i

Na prática pode-se colocar a coluna de custo reduzido mais negativo de cada agente para acelerar o algoritmo.

Estabilização da Geração de Colunas

- A geração de colunas como descrita por (Savelsberg, 1999) demora a convergir
- As variáveis duais oscilam muito
- Numa iteração algumas variáveis duais exageradamente grandes fazem com que alocar certas tarefas a certos agentes seja altamente atraente. Em consequência o subproblema gera colunas com estas tarefas para estes agentes.
- Entretanto estas colunas extremas dificilmente serão as colunas boas para a solução final da geração de colunas.

Estabilização da Geração de Colunas

Seja um programa linear P viável e limitado e o seu dual D

$$\begin{array}{ll|ll} \min & cx & \max & b\pi \\ (P) \quad s.a & Ax = b & (D) \quad s.a. & \pi A \leq c \\ & x \geq 0 & & \end{array}$$

Estabilização da Geração de Colunas

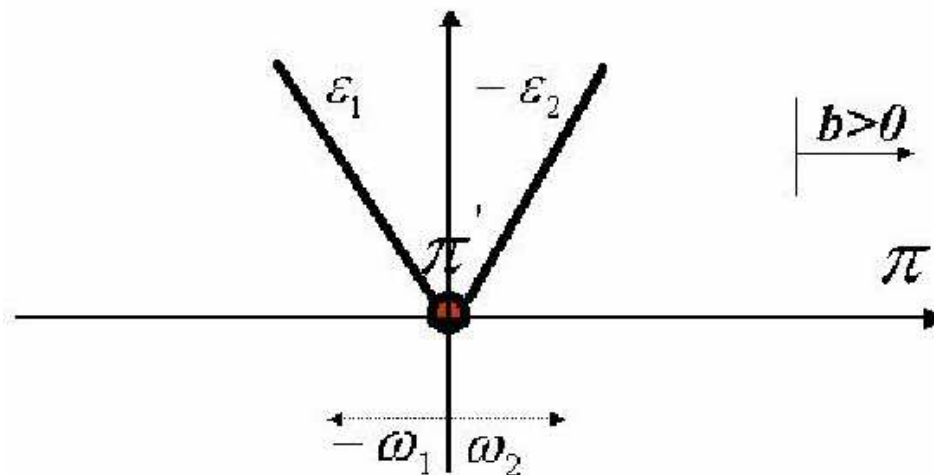
Redução da degenerescência feita perturbando-se P

$$\begin{aligned} & \min \quad cx \\ (P_p) \quad & s.a. \quad Ax - y_1 + y_2 = b \\ & \quad \quad x \geq 0 \\ & \quad \quad 0 \leq y_1 \leq \epsilon_1, 0 \leq y_2 \leq \epsilon_2 \end{aligned}$$

Este processo evita que as variáveis duais oscilem.

Estabilização da Geração de Colunas

$$\begin{aligned}
 & \max \quad b\pi - \omega_1 \epsilon_1 - \omega_2 \epsilon_2 \\
 (D_p) \quad & s.a. \quad \pi A \leq c \\
 & \omega_2 \geq \pi \\
 & \omega_1 \geq -\pi
 \end{aligned}$$



Estabilização da Geração de Colunas

Outra maneira de reduzir a oscilação é explicitamente limitar o valor das variáveis duais a um intervalo.

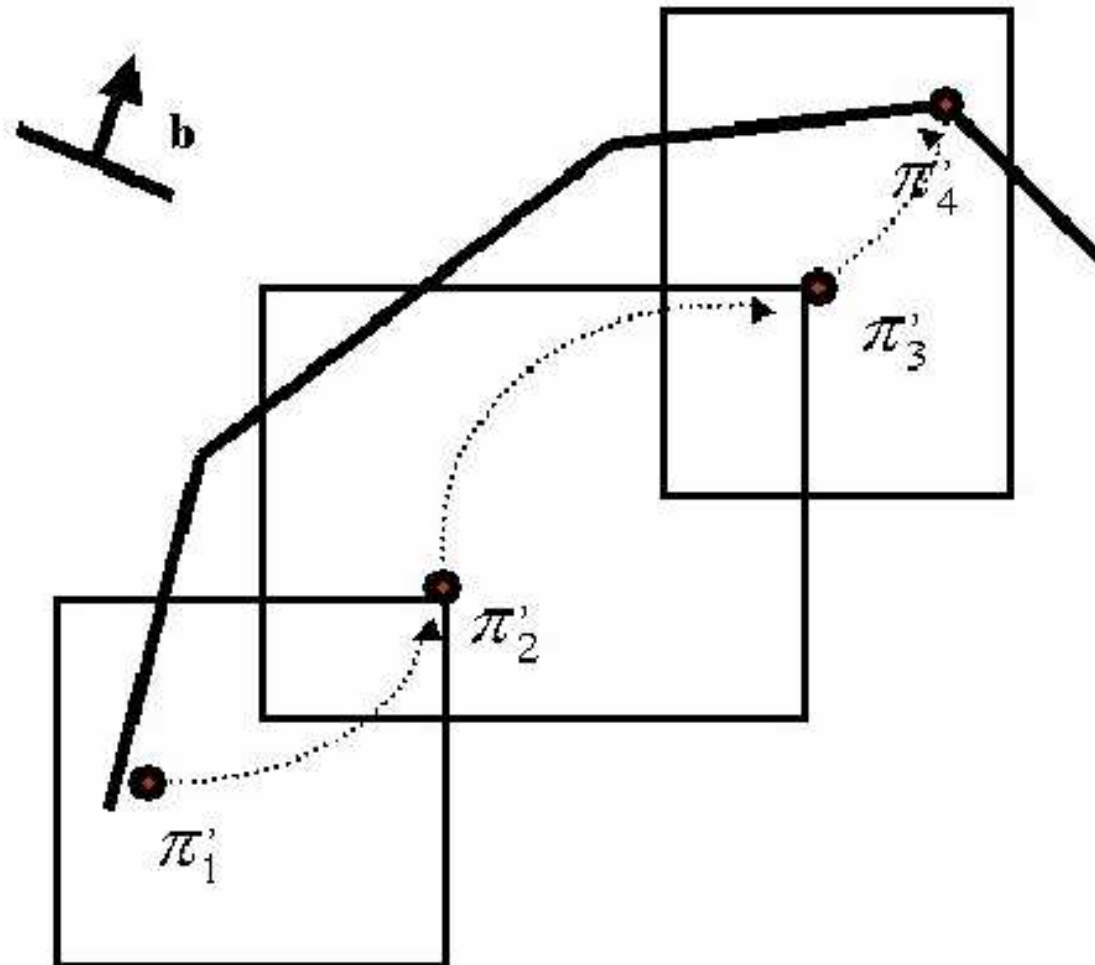
$$\begin{aligned} \max \quad & b\pi \\ (D_r) \quad & s.a. \quad \pi A \leq c \\ & d_1 \leq \pi \leq d_2 \end{aligned}$$

Estes limites no dual correspondem ao seguinte problema primal:

$$\begin{aligned} \min \quad & cx - d_1y_1 + d_2y_2 \\ (P_d) \quad & s.a. \quad Ax - y_1 + y_2 = b \\ & x \geq 0 \\ & y_1 \geq 0, y_2 \geq 0 \end{aligned}$$

Estabilização da Geração de Colunas

Algoritmo do Passo da Caixa



Estabilização da Geração de Colunas

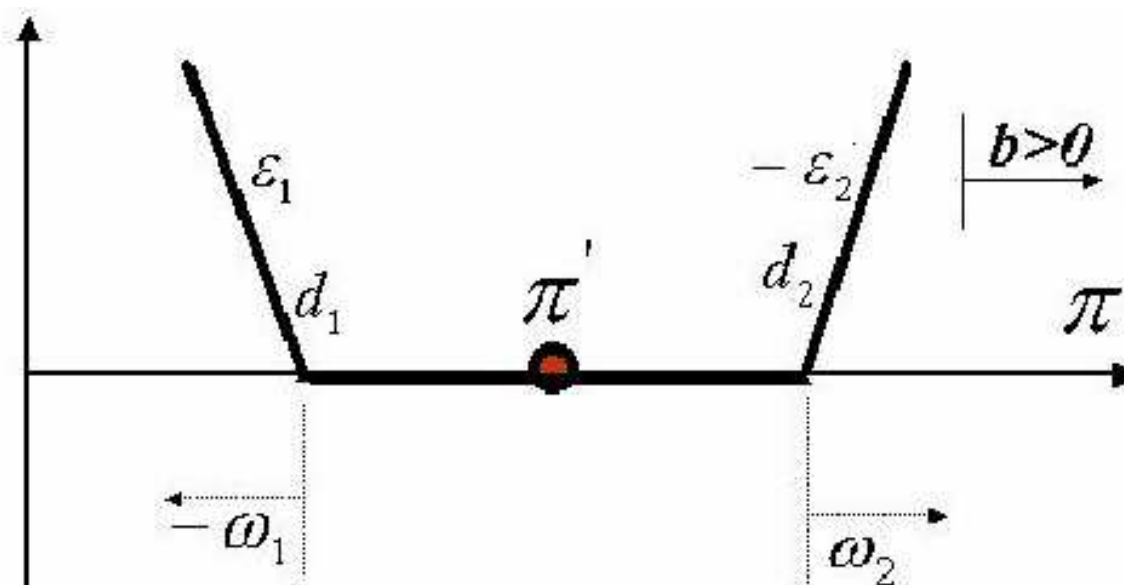
A junção das formulações P_p e P_d resulta na formulação do método de estabilização da geração de colunas P_e :

$$\begin{array}{llllll}
 \min & cx & -d_1y_1 & +d_2y_2 & & \\
 (P_e) \quad s.a & Ax & -y_1 & +y_2 & = & b \\
 & & y_1 & & \leq & \epsilon_1 \\
 & & & y_2 & \leq & \epsilon_2 \\
 & x \geq 0, & y_1 \geq 0, & y_2 \geq 0 & &
 \end{array}
 \quad \left| \quad \begin{array}{l}
 \pi \\
 -\omega_1 \leq 0 \\
 -\omega_2 \leq 0
 \end{array} \right.$$

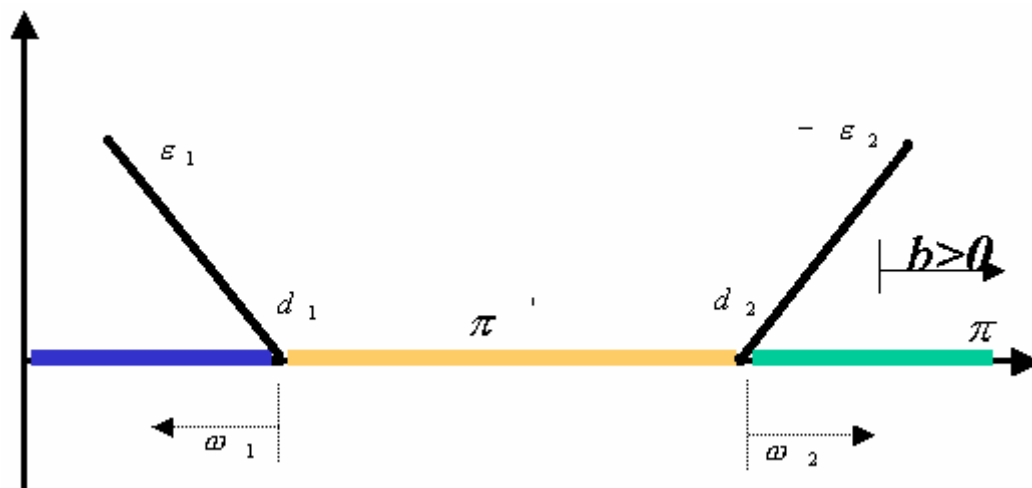
Estabilização da Geração de Colunas

O problema dual associado a P_e é como segue:

$$\begin{array}{ll}
 \max & b\pi - \omega_1 \epsilon_1 - \omega_2 \epsilon_2 \\
 (D_e) \quad \text{s.a.} & \pi A \leq c \\
 & d_1 - \omega_1 \leq \pi \leq d_2 + \omega_2 \\
 & \omega_1 \geq 0, \omega_2 \geq 0
 \end{array}$$



Estabilização da Geração de Colunas



Se o valor dual é pequeno, recentralizar e aumentar o intervalo

Se o valor dual está dentro do intervalo, recentralizar e reduzir o intervalo

Se o valor dual é grande, recentralizar e aumentar o intervalo

Estabilização da Geração de Colunas

- Esse algoritmo proposto por (Du Merle, Desrosiers e Hansen, 1999) é complicado porque é necessário estimar 4 parâmetros d_1, d_2, ϵ_1 e ϵ_2 viável dual:
- Não é possível determinar quantas iterações serão necessárias.

Estabilização da Geração de Colunas

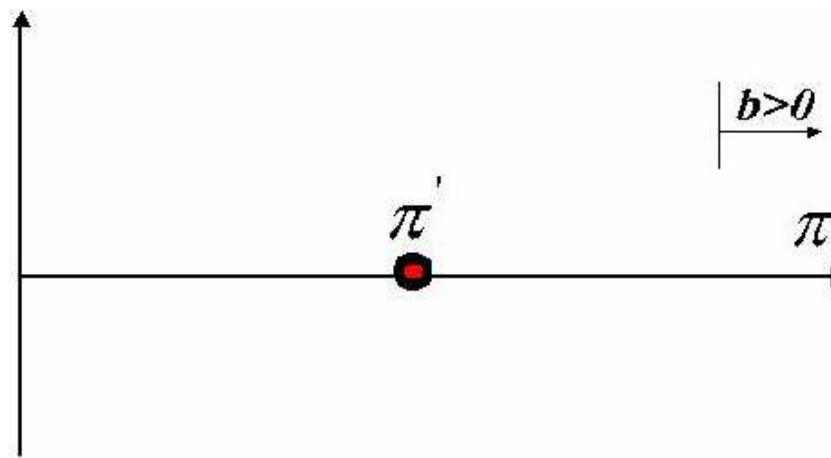
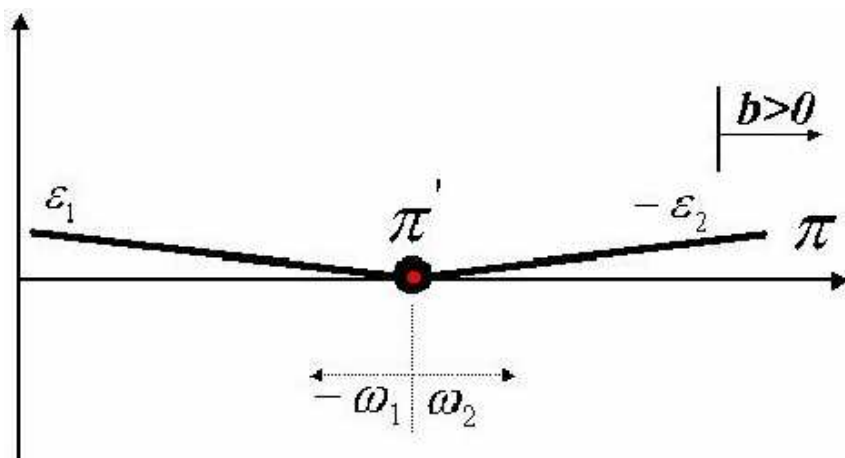
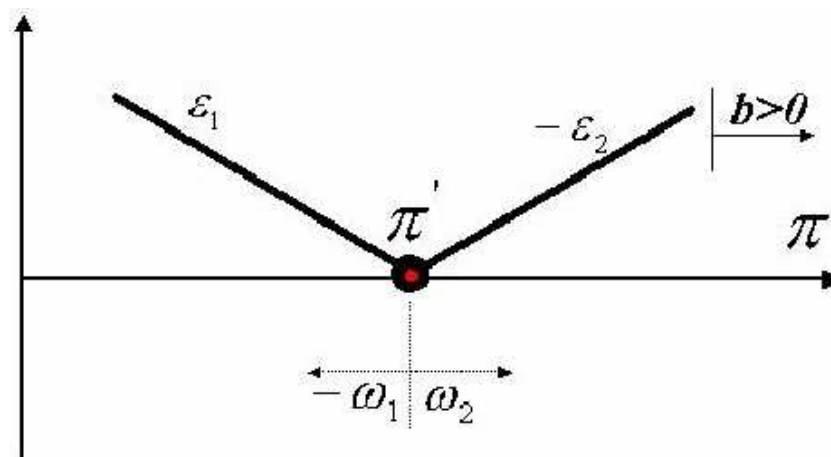
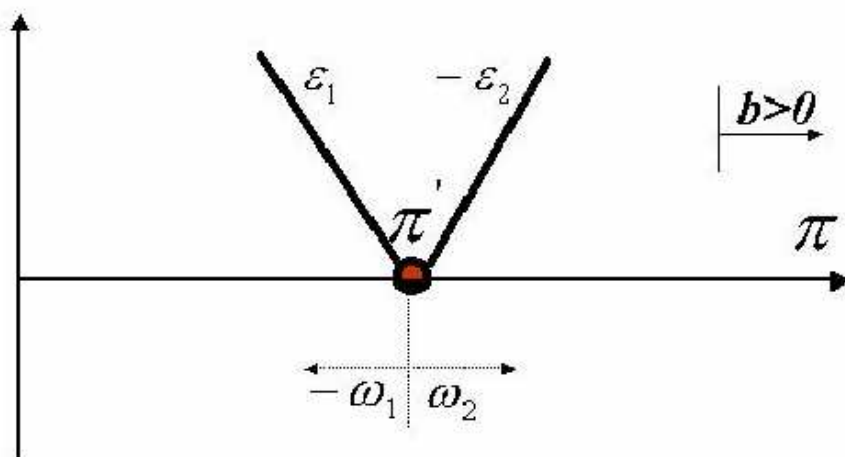
Nós propomos um algoritmo simplificado:

- O intervalo $[d_1, d_2]$ é reduzido a um ponto π' e $\epsilon_1 = \epsilon_2 = \epsilon$.
- Iniciamos π' com a solução dual ótima da formulação clássica (PAG-C).
- O algoritmo sempre termina em 4 iterações

Estabilização da Geração de Colunas

Neste trabalho fez-se $d_1 = d_2 = \pi'$

e executa-se 4 iterações com os respectivos valores para $\epsilon \in 0.1, 0.01, 0.001$ e 0 .



Estabilização da Geração de Colunas

```
 $\pi$  = duais da Relaxação;  
 $\epsilon = 0.1$ ;  
valRetorno = geraColunas();  
if valRetorno  $\geq$  cutoff  $- 1 + \epsilon$  then  
   $\perp$  return valRetorno  
  
for  $i = 0; i < 3; i++$  do  
   $\pi$  = novos duais calculados pela geraColunas;  
  if  $i == 2$  then  
     $\epsilon = 0$ ;  
  else  
     $\perp$   $\epsilon = 0.1^{i+2}$ ;  
  valRetorno = geraColunas();  
  if valRetorno  $\geq$  cutoff  $- 1 + \epsilon$  then  
     $\perp$  return valRetorno  
  
return valRetorno
```

Efeito da Estabilização

Tempos gastos para resolver o nó raiz, antes e depois da estabilização da geração de colunas.

tipo	m	n	LB	melhor conhecido	tempo antes estabilização	tempo depois estabilização
D	5	100	6350	6353	18.71	3.59
D	5	200	12741	12743	6610.31	138.03
D	10	100	6342	6349	1.98	0.78
D	10	200	12426	12433	181.25	19.23
D	20	100	6177	6196	0.79	0.53
D	20	200	12230	12244	26.12	5.01

Branch-and-Price para o PAG

- Após o procedimento de geração de colunas ter terminado, somente no caso das variáveis y_i^k serem inteiras é que a solução para o problema mestre será a solução para o problema inteiro original.
- Caso contrário se faz necessário a execução de um procedimento de *branch-and-bound* para se obter a solução inteira do problema mestre

Branch-and-Price para o PAG

- Deve ser escolhida uma regra de ramificação que seja compatível com o subproblema, para evitar que colunas excluídas pela regra de ramificação sejam geradas novamente.
- Toda solução viável da formulação (PAG-Exp) tem uma solução viável correspondente para a formulação (PAG-C). Então a idéia é fazer as ramificações usando a formulação (PAG-C) enquanto trabalha-se com a (PAG-Exp)

Branch-and-Price para o PAG

Na formulação (PAG-C):

- Fixar x_{ij} a 0 proíbe a tarefa j de ser alocada ao agente i .
- Fixar a variável x_{ij} a 1 exige que a tarefa j seja alocada ao agente i .

Branch-and-Price para o PAG

Na formulação (PAG-Exp):

- Para proibir que uma tarefa j seja alocada ao agente i , se $v_{ij}^k = 1$, então faz-se $y_i^k = 0$ para todo $k \in K_i$.
- Para assegurar que uma tarefa j seja alocada ao agente i , se $v_{ij}^k = 0$, então faz-se $y_i^k = 0$ para todo $k \in K_i$, e se $v_{lj}^k = 1$, faz-se $y_l^k = 0$ para $1 \leq l \neq i \leq m$ e $k \in K_l$.

Branch-and-Price para o PAG

Sejam:

- J_i^0 o conjunto de todos os índices j tais que x_{ij} foi fixado à 0; e
- J_i^1 o conjunto análogo para fixação à 1.
- $J_{\bar{i}}^1$ o conjunto de todas as tarefas que foram fixadas aos agentes diferentes de i , ou seja j tais que $x_{i'j}$ foi fixado a 1 para $i \neq i'$.

O subproblema de geração de colunas que leva em conta estas fixações fica então:

$$\max \sum_{j \in \{1, \dots, n\} \setminus (J_i^0 \cup J_i^1 \cup J_{\bar{i}}^1)} (u_j - c_{ij}) \alpha_{ij} + \sum_{j \in J_i^1} (u_j - c_{ij})$$

s.a.

$$\sum_{j \in \{1, \dots, n\} \setminus (J_i^0 \cup J_i^1 \cup J_{\bar{i}}^1)} a_{ij} \alpha_j^i \leq b_i - \sum_{j \in J_i^1} a_{ij}$$

$$\alpha_j^i \in \{0, 1\} \quad j \in \{1, \dots, n\}$$

Branch-and-Price para o PAG

```
valSolucao = geraColunasEstabilizada(valMelhorSolucao);  
if valSolucao  $\geq$  valMelhorSolucao - 1 +  $\epsilon$  then  
  | return;  
  
determinaProximoFixado(i,j);  
if i == 0 & & j == 0 then  
  | //Solução Inteira Encontrada  
  | if !usandoColunas() then  
  |   | if valSolucao < valMelhorSolucao then  
  |   |   | valMelhorSolucao = valSolucao;  
  |   |  
  |   | return;  
  |  
  | //fixando em 0  
  | retirarColunasViolamZero(i,j);  
  | fixarZero(i,j);  
  | chamarAlgoritmoRekursivamente();  
  | recolocarColunasViolamZero(i,j);  
  | desFixarZero(i,j);  
  | //fixando em 1  
  | retirarColunasViolamUm(i,j);  
  | fixarUm(i,j);  
  | chamarAlgoritmoRekursivamente();  
  | recolocarColunasViolamUm(i,j);  
  | desFixarUm(i,j);
```

Resolução Exata do PAG

Resultados Computacionais

tipo	m	n	LB	melhor conhecida	algoritmo deste trabalho	tempo melhor solução	tempo para provar ótimo	nó melhor solução	total nós visitados
C	5	100	1930	†1931	*1931	0.93	1.17	3	5
C	5	200	3455	†3456	*3456	420.38	422.98	46	47
C	10	100	1400	†1402	*1402	1.12	1.84	9	17
C	10	200	2804	†2806	*2806	224.78	266.17	22	35
C	20	100	1242	†1243	*1243	1.92	2.12	25	29
C	20	200	2391	†2391	*2391	53.81	55.38	22	23
D	5	100	6350	†6353	*6353	61.74	96.30	106	171
D	5	200	12741	12743	**†12742	312.68	583.68	11	57
D	10	100	6342	6349	**†6347	705.76	818.62	2416	2687
D	10	200	12426	12433					
D	20	100	6177	6196					
D	20	200	12230	12244					
E	5	100	12673	†12681	*12681	27.68	30.09	57	63
E	5	200	24927	†24930	*24930	1080.56	1305.62	27	31
E	10	100	11568	†11577	*11577	12.53	22.85	47	87
E	10	200	23302	†23307	*23307	135.62	670.62	19	155
E	20	100	8431	†8436	*8436	6.60	6.81	36	37
E	20	200	22377	†22379	*22379	36.79	40.74	18	21

tipo	m	n	LB		tempo
			antigo	novο	
D	10	200	12426	12430	47197.26
D	20	100	6177	6180	30528.46
D	20	200	12230	12234	38469.81

Branch-and-Price para o PAG

- A instância D20_100 foi posteriormente resolvida em 2807 nós e 1043s, graças a uma heurística (baseada em GC) que melhorou o melhor UB de 6196 para 6185 (que era o ótimo).

FIM

Geração de Colunas para Programação Inteira: Parte III

Eduardo Uchoa

Dep. Engenharia de Produção

Universidade Federal Fluminense

ELAVIO 2007

Conteúdo da Parte III

- Recentemente descobriu-se maneiras eficientes de combinar a geração de colunas com uma outra técnica fundamental em PI, a separação de cortes.
- Os chamados algoritmos de *branch-cut-and-price* tem se mostrado muito poderosos e obtiveram grandes progressos na resolução exata de vários problemas clássicos.
- Esse é um tema de pesquisa bastante atual.

Conteúdo da Parte III

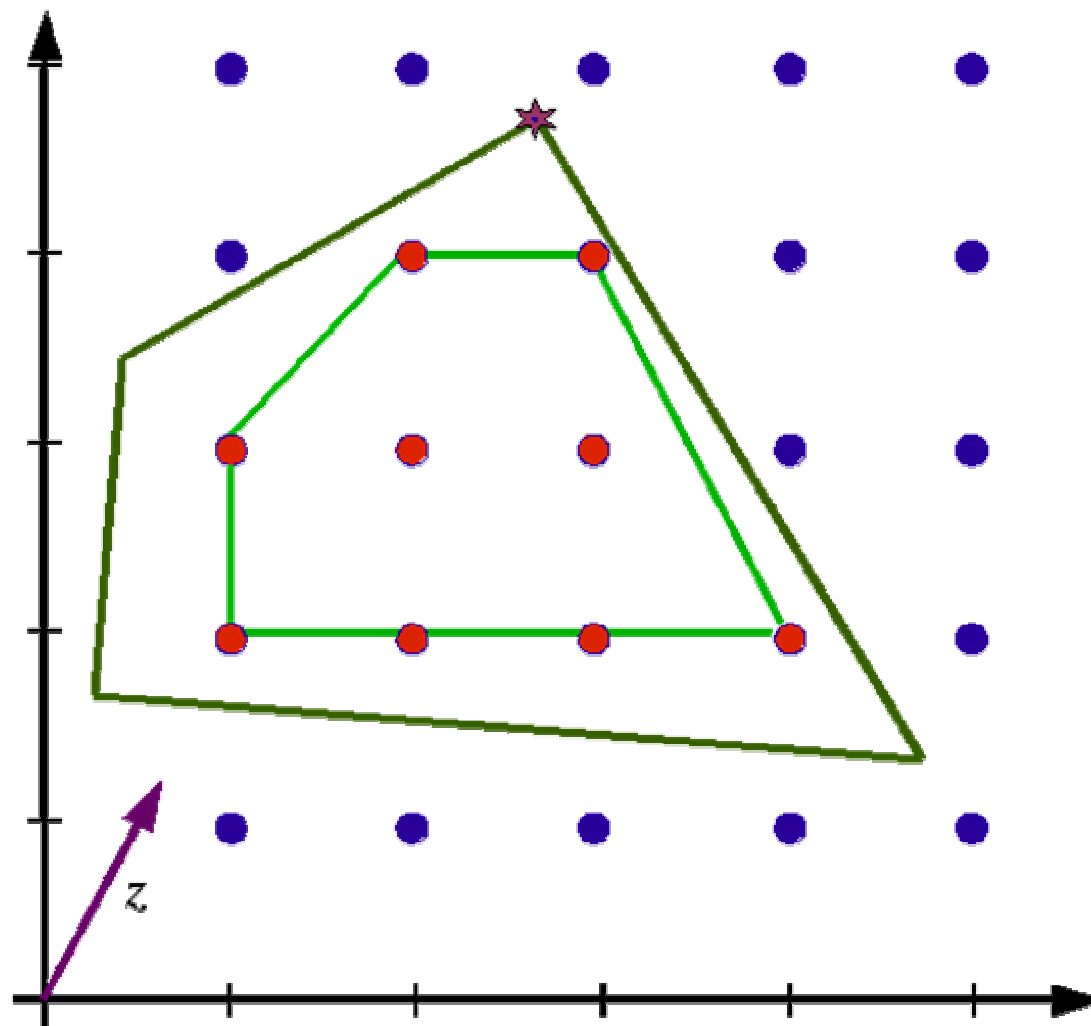
1. Combinando geração de colunas com separação de cortes.
2. Branch-cut-and-price *Robusto*
3. Casos de sucesso
4. Branch-cut-and-price *Robusto* sobre formulações extendidas

Algoritmo de planos de corte

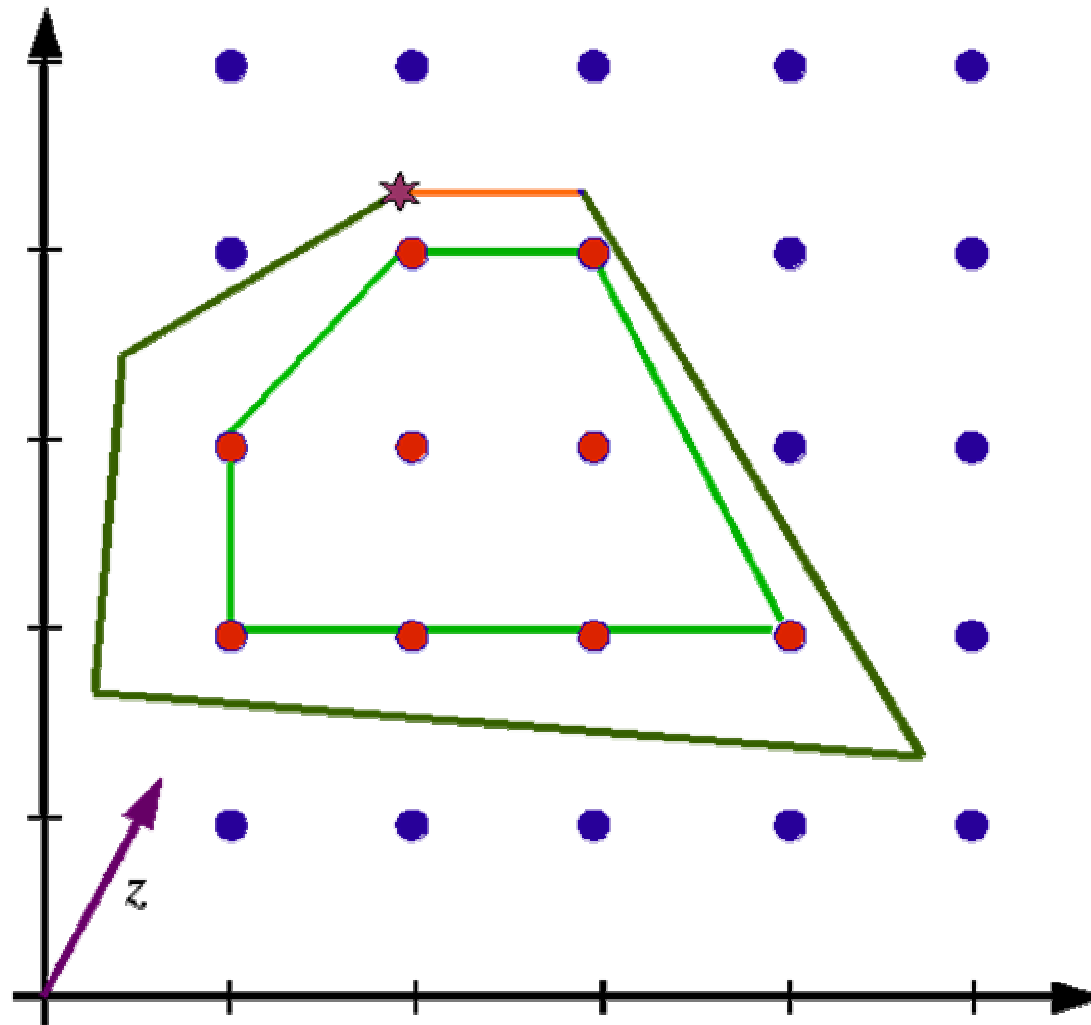
Uma outra maneira de reforçar uma formulação é através da adição de cortes

- Dada uma solução fracionária de uma formulação, busca-se adicionar uma nova desigualdade que corte essa solução, mas não corte nenhuma solução inteira.
- Existe intensa pesquisa sobre técnicas para de gerar bons cortes, o que se chama de *algoritmos de separação*.

Solução fracionária



Nova solução fracionária após a separação de um corte



Branch-and-cut

- A combinação de separação de cortes com a técnica de branch-and-bound gerou os chamados algoritmos de branch-and-cut.
- Esse tipo de algoritmo também surgiu nos anos 80 e hoje é o método mais usado para resolver PIs de forma exata.

Branch-cut-and-price

Naturalmente se considerou a possibilidade de combinar a geração de colunas com a separação de cortes para produzir algoritmos mais poderosos.

O problema é que os cortes naturais introduzidos no mestre sobre as variáveis λ alteram a estrutura do pricing, tendendo a o tornar intratável.

A primeira tentativa de um BCP foi feita para o problema de coloração de arestas (Nemhauser e Park, 1991).

Branch-cut-and-price robusto

Um BCP foi definido como *robusto* apenas se tanto as operações de branching quanto a separação de cortes jamais alterarem a estrutura do subproblema de pricing ao longo do algoritmo (Poggi de Aragão e Uchoa, 2003).

Equívoco encontrado na literatura

Alguns autores afirmam que o branch-and-price nada mais é que o “dual” do branch-and-cut: o pricing de colunas equivale a separação de cortes no PL dual.

Realidade

Existe uma diferença fundamental:

- Se o subproblema de separação for NP-difícil e intratável na prática, pode-se usar heurísticas. Alguns cortes violados podem ser perdidos, mas um limite válido é obtido.
- Se o subproblema de pricing for NP-difícil e intratável na prática, mesmo com boas heurísticas, é preciso resolver exatamente pelo menos uma vez para obter um limite válido.

Realidade

- Se em algum momento o pricing não puder mais ser resolvido de forma exata, o algoritmo inteiro falha !
- Isso explica porque a definição de branch-cut-and-price *robusto* exige que a separação não atrapalhe o pricing, mas não o contrário.

Branch-cut-and-price robusto

Alguns pesquisadores notaram que cortes expressos em termos das variáveis de certas formulações originais podiam ser adicionados sem alterar o pricing e construíram os primeiros BCPs robustos:

Vanderbeck (1998), dimensionamento de lotes.

Kim, Barnhart, Ware and Reinhardt (1999), um problema de projeto de redes.

Kohl, Desrosiers, Madsen, Solomon and Soumis (1999), Roteamento de veículos com janelas de tempo

Van der Akker, Hurkens and Savelsbergh (2000), um problema de scheduling.

Barnhart, Hane and Vance (2001), multi-fluxo inteiro.

Poggi de Aragão e Uchoa (2003)

- Proposta de um esquema geral de reformulação para permitir BCP robusto em problemas onde antes isso não era possível (Ex: bin packing e graph coloring)
- Técnica para obter os custos reduzidos das variáveis originais a partir do PL mestre.
- Experimentos em problemas clássicos mostrando o potencial do BCP robusto.

Capacitated Vehicle Routing Problem (Fukasawa et al. 2003, 2006).

Instance

Undirected graph $G=(V,E)$, $V= \{0,1,\dots,n\}$, $|E|=m$,

Vertex 0 represents a depot and remaining vertices represent clients

Edge lengths, denoted by $c(e)$

Client demands, denoted by $q(i)$

Number K of vehicles and vehicle capacity C

Solution

A route for each of the K vehicles satisfying the following constraints:

- (i) Routes start and end at the depot,
- (ii) each client is visited by exactly one vehicle
- (iii) the total demand of clients visited in a route is at most C .

Goal

Minimize total route length.

Capacitated Vehicle Routing Problem (CVRP)

Hard to solve to optimality

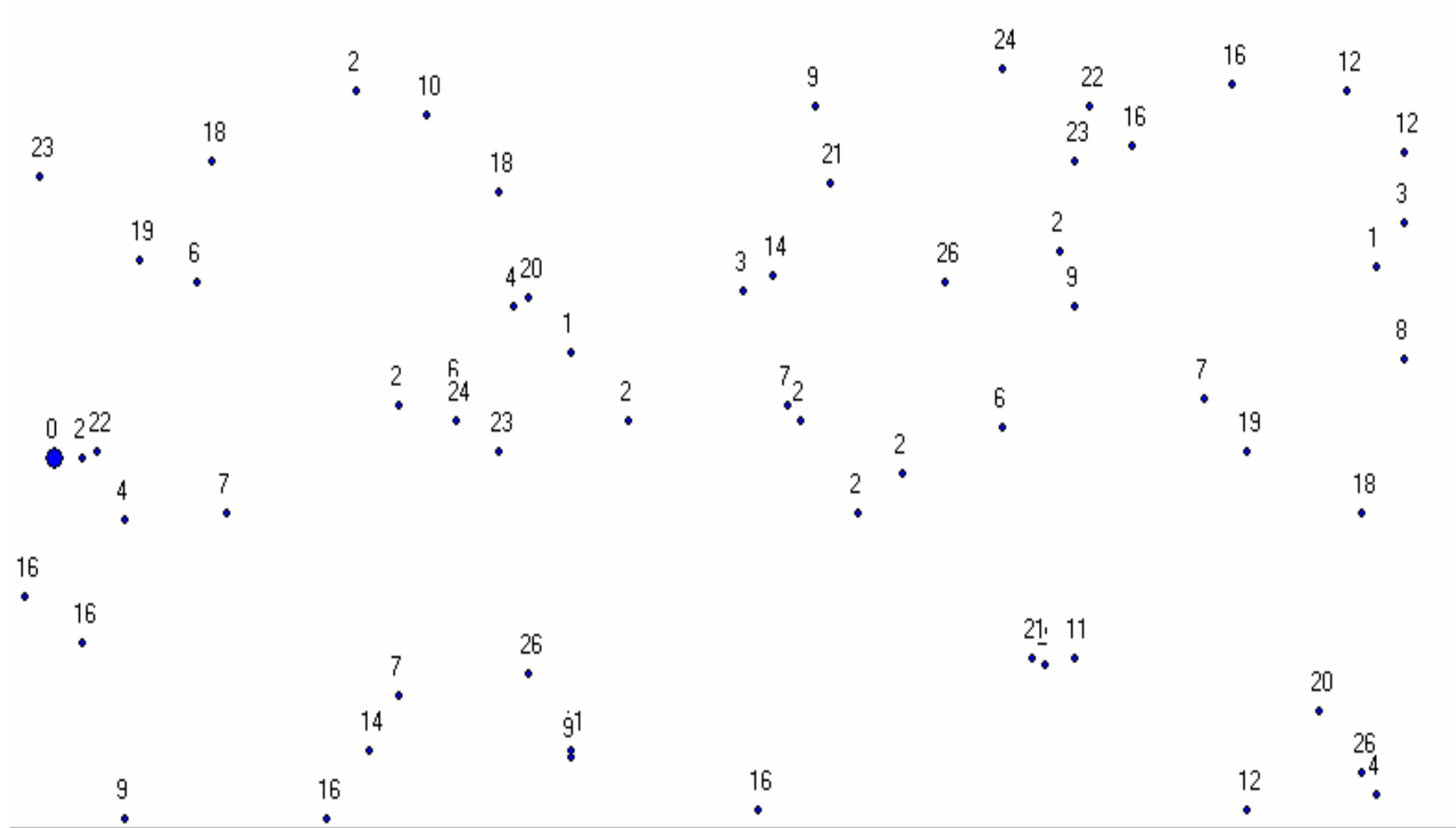
The evolution of the algorithms has been very slow:

State of the art in 1978: Consistent solution for up to 25 clients.

State of the art in 2003: Consistent solution for up to 50 clients.

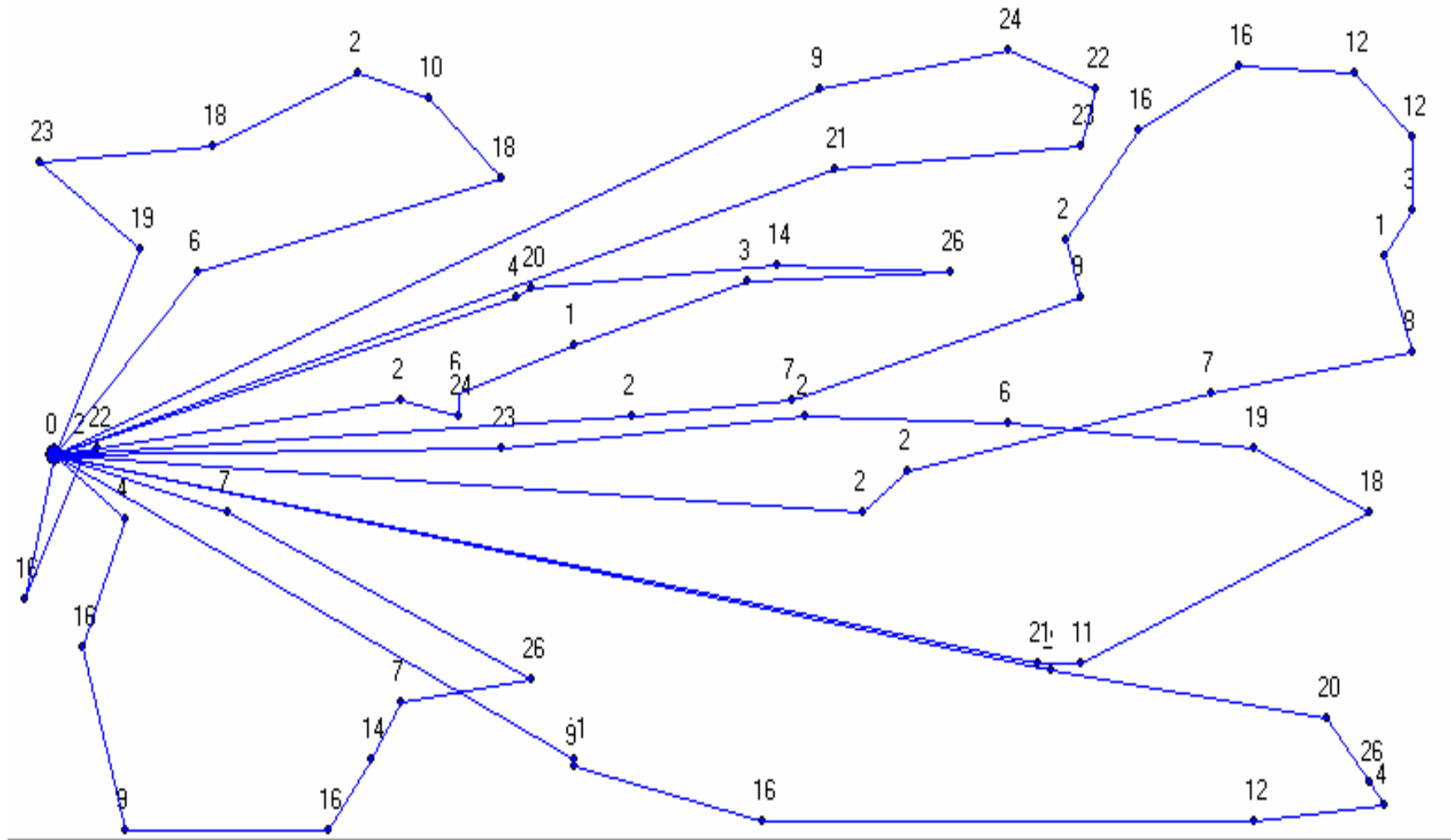
This work solved all instances from the literature with up to 135 clients.

Example - Instance A-n62-k8



Vehicle Capacity: 100

Optimal Solution A-n62-k8



Optimal Solution Value: 1288

BCP robusto para o CVRP

Subproblema de pricing: caminho mais curto com capacidade e eliminação de s-ciclo. Resolvido por programação dinâmica com complexidade $O(s!mC)$.

Separação de cortes de Rounded Capacity (RC), strengthened combs, multistars, framed capacities e hypotours (cortes específicos para o CVRP e já conhecidos na literatura).

E series (Eilon-Christofides, 1969)

Instance	Root Lower Bound		Upper Bound		Total Time (s)
	LLE03	Ours	Previous	Ours	
E-n51-k5	519.0	519.1	521	521	65
E-n76-k7	666.4	670.8	682	682	46520
E-n76-k8	717.9	726.8	735	735	22891
E-n76-k10	799.9	817.4	830	830	80722
E-n76-k14	969.6	1006.5	1021	1021	48637
E-n101-k8	802.6	805.2	815	815	80963
E-n101-k14	1026.9	1053.9	1071	1067	116284 = 32h

Root gaps over 41 hard instances between 50 and 135 clients

	Avg. Gap (%)
Only RC cuts (exact separation)	2.92
Only column generation ($s = 2$)	4.76
RC cuts + column generation ($s = 2$)	1.02
LLE03 (best BC in the literature)	2.26
All cuts + column generation ($s = 2$)	0.97
All cuts + column generation ($s = 3$)	0.82
All cuts + column generation ($s = 4$)	0.74

Programação de Horários em Escolas – *School Timetabling*

- Alocar um conjunto de aulas (encontros entre professores e turmas) em uma grade de horários semanal, cumprindo algumas regras obrigatórias e, dentro do possível, satisfazendo preferências pedagógicas, pessoais e institucionais.

Não ocorrência de conflitos





		Segunda				
Turma	Dia, Per.	1º Per.	2º Per.	3º Per.	4º Per.	5º Per.
						1º Per.
101 A		Prof. João	Prof. João	Profa. Soraia	Prof. Alberto	Profa. Júlia
101 B		Profa. Soraia	Prof. Alberto	Prof. João	Profa. Júlia	Profa. Regina
201 A		Profa. Júlia	Profa. Júlia	Prof. Carlos	Prof. João	Profa. Soraia
201 B		Profa. Regina	Prof. Soraia	Prof. Alberto	Prof. Carlos	Prof. Carlos
201 C		Prof. Carlos	Prof. Carlos	Profa. Júlia	Profa. Regina	Prof. João
301 A		Prof. Alberto	Profa. Regina	Profa. Regina	Profa. Soraia	Prof. Alberto
301 B		Prof. Celso	Prof. Celso	Prof. João	Prof. Artur	Prof. Artur
301 C		Prof. Artur	Prof. Artur	Prof. Celso	Prof. Celso	Prof. Antônio

Distribuição de Aulas

<div> <div></div> <div>Dia</div> </div> <div> <div>Período</div> <div></div> </div>	Segunda	Terça	Quarta	Quinta	Sexta
1	Prof. João	Prof. Alberto	Profa. Júlia	Profa. Júlia	Profa. Soraia
2	Prof. João	Prof. Alberto	Profa. Júlia	Profa. Júlia	Profa. Soraia
3	Prof. Carlos	Profa. Soraia	Prof. João	Profa. Joana	Prof. João
4	Prof. Carlos	Profa. Soraia	Prof. João	Profa. Joana	Prof. Carlos
5	Profa. Júlia	Profa. Joana	Prof. Carlos	Prof. Alberto	Prof. Alberto



- Evitar excesso de aulas com o mesmo professor/disciplina na mesma turma em um determinado dia
- Dentro do possível oferecer **aulas geminadas** para certos encontros entre prof. e turmas

Preferências do Professor

Período \ Dia	Segunda	Terça	Quarta	Quinta	Sexta
1	300A				100C
2	300A				100C
3					
4			300A		
5			200B	200B	
<i>Dias com atividade</i>    					

- Tentar agrupar a maioria das aulas em pouco dias de trabalho

Preferências do Professor

<div> <div></div> <div></div> </div>	Dia	Segunda	Terça	Quarta	Quinta	Sexta
Período						
1		300A				
2		300A	300A			
3		100C				
4		100C	200B			
5		200B				
Dias com atividade						

- Evitar “buracos” em dias de trabalho

Branch-Cut-and-Price (Santos, Ochi e Uchoa, 2007).

Período \ Dia	Dia Letivo
1	300A
2	300A
3	
4	
5	200B

Coluna: variáveis representam diferentes padrões de alocação em um dia de trabalho do professor

Cortes específicos e cortes de *knapsack cover* considerando o fechamento da carga horária total de professores e de professores em turmas

Resultados Obtidos

Instância	Aulas	LB Form. Compacta + Cortes	LB Formulação Ger. Col. e Cortes	Melhor UB
1	75	189	202	202
2	150	333	333	333
3	200	414	423	426
4	300	643	652	653
5	325	756	762	766
6	350	738	756	778
7	500	999	1017	1045

BCP robusto sobre formulações extendidas

Nova linha de pesquisa (desde 2005) que tenta aproveitar ao máximo o potencial do BCP robusto.

Ao invés de apenas misturar a geração de colunas com cortes sobre formulações tradicionais, tirar proveito da maneira que o pricing está sendo resolvido (tipicamente por programação dinâmica) para gerar novas famílias de cortes sobre formulações extendidas de tamanho muito grande.

Esses novos cortes são garantidamente mais fortes do que os cortes tradicionais.

The Capacitated Minimum Spanning Tree Problem (CMST)

■ Given:

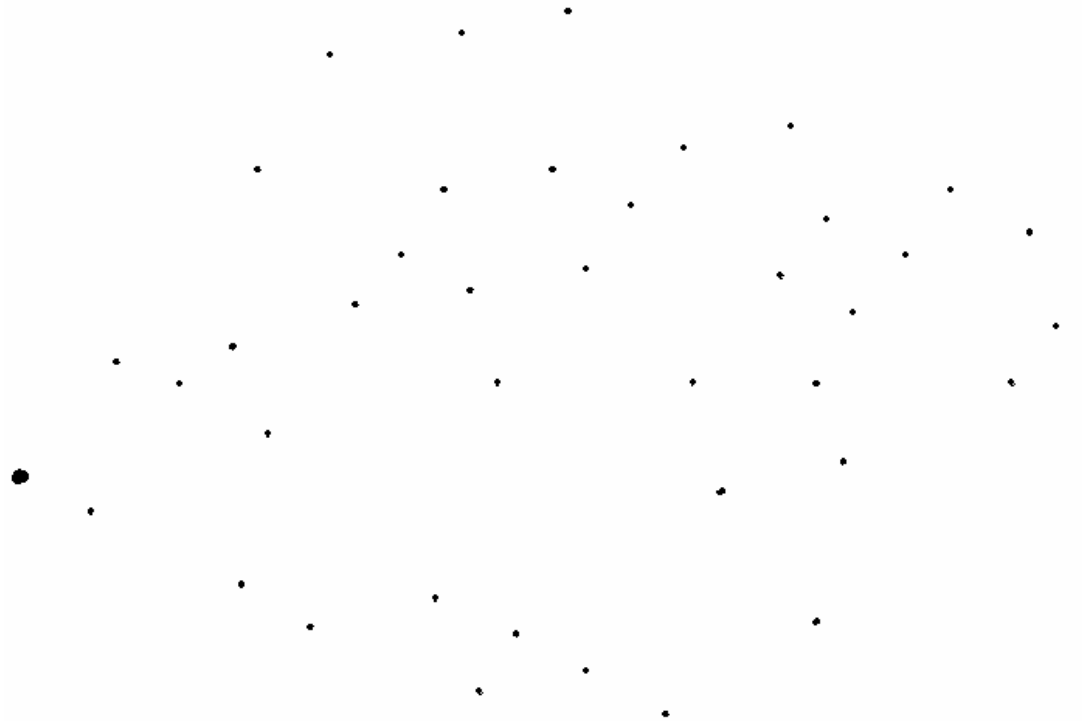
- An undirected graph $G=(V,E)$,
- Costs c_e for each edge in E ,
- Non-negative integral weights (or demands) d_v for each vertex in V ,
- A maximum capacity C ,
- A vertex designed as “the root”.

■ Find:

- A minimum cost spanning tree where the total weight of the vertices in each subtree hanging from the root does not exceeds C .

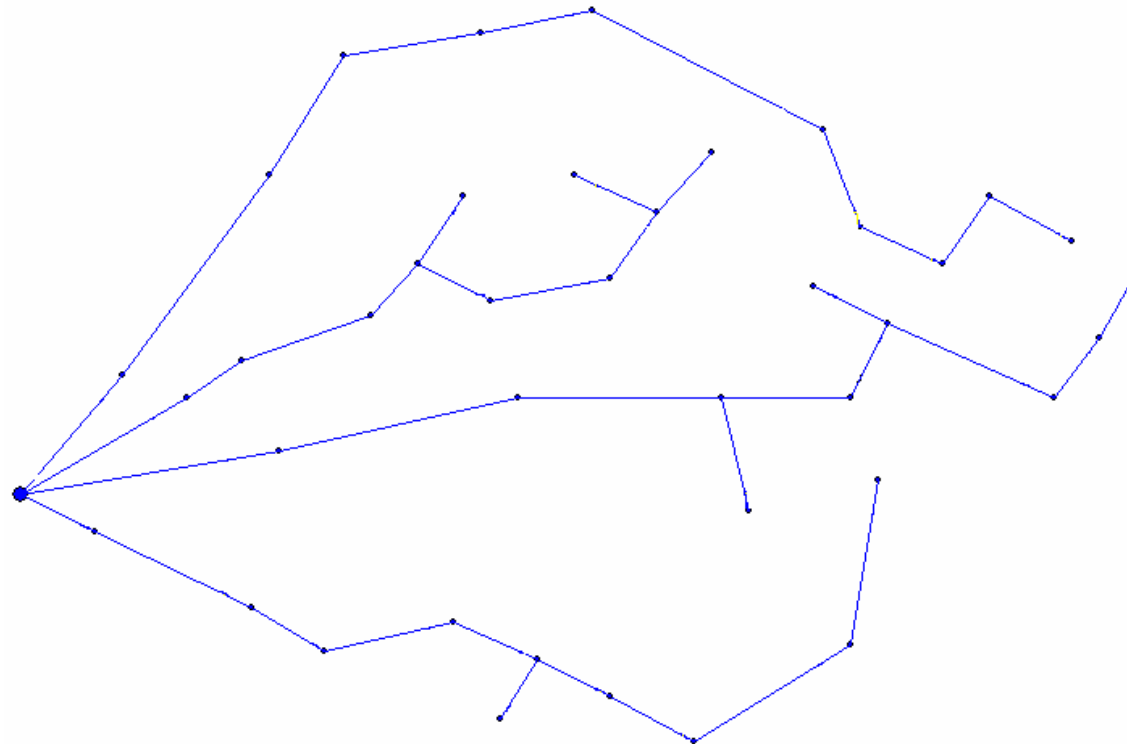
The Capacitated Minimum Spanning Tree Problem (CMST)

- Example: (unit demands, $C=10$)



The Capacitated Minimum Spanning Tree Problem (CMST)

- Example: (unit demands, $C=10$)



A formulation for the CMST on a directed graph

- Replace each edge $(i,j) \in E$ by two arcs (i,j) and (j,i) , with cost $c_{ij} = c_{ji} = c_e$
- Let 0 be the root node
- Let $V_+ = \{1, \dots, n\}$
- Define binary variables
 - $X_a = 1$ if arc a belongs to an arborescence rooted at 0.

A Formulation for the CMST

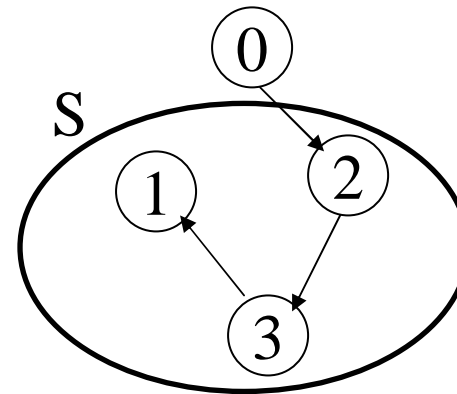
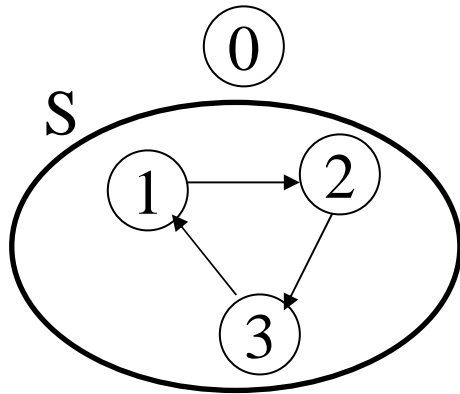
$$\begin{aligned} \min \quad & \sum_{a \in A} c_a x_a \\ \text{s.t.} \quad & \sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V_+ \\ & \sum_{a \in \delta^-(S)} x_a \geq \left\lceil \frac{d(S)}{C} \right\rceil, \forall S \subseteq V_+ \\ & x_a \in \{0,1\}, \forall a \in A \end{aligned}$$

Ensures that all vertices have exactly one arc incident

Capacity Cut (CC):
Eliminates subtours and reinforces maximum tree weight

Violated Capacity Cuts

C=2, unit demands, root = 0



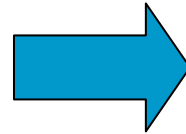
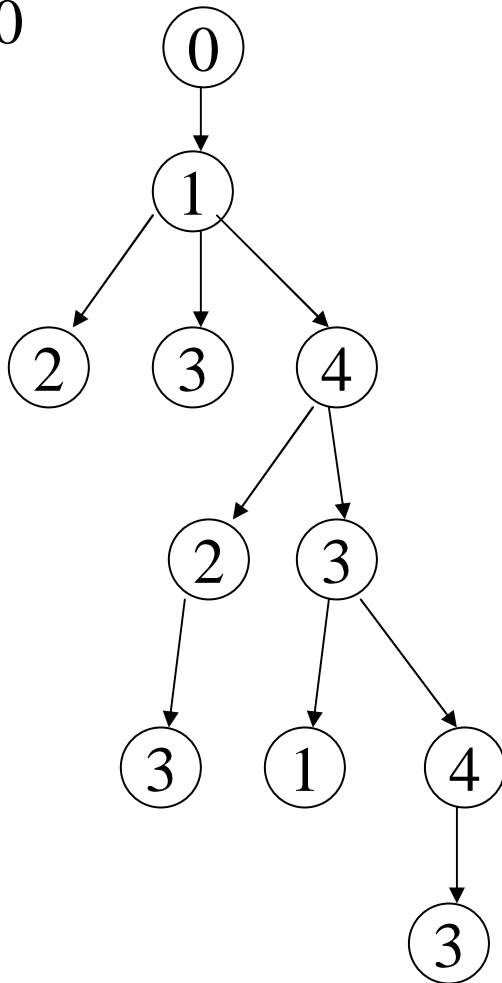
$$\left\lceil \frac{d(S)}{C} \right\rceil = 2 > \sum_{a \in \delta^-(S)} x_a = 1$$

BCP for the CMST

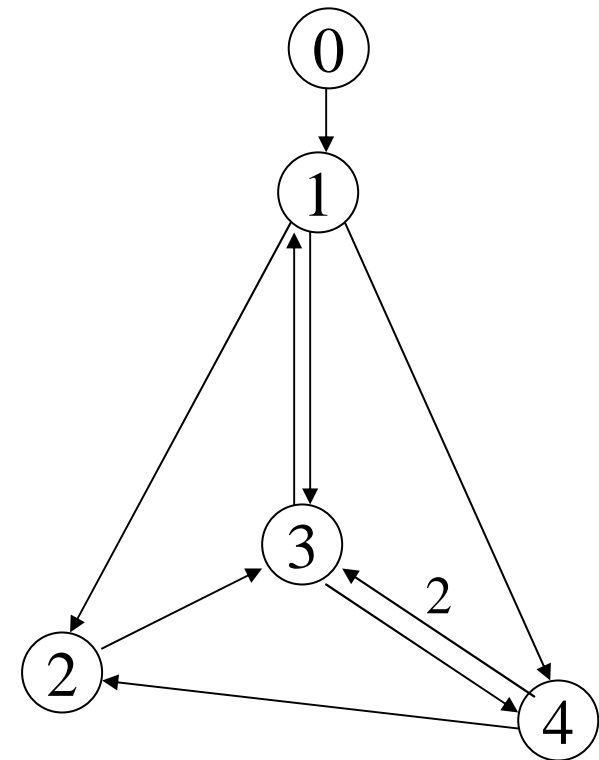
- Work on the directed graph,
- Columns are associated to q-arbs,
- Valid cuts over the arc formulation can be added, including Capacity Cuts.

q-arborescences (q-arbs)

Example of a q-arb with cycles,
 $C=10$



Which corresponds to:



Formulation combining q-arbs with CCs

$$\begin{aligned}
 & \min && \sum_{a \in A} c_a x_a \\
 & s.t. && \sum_{t \in T: a \in t} q_t^a \lambda_t - x_a = 0, \forall a \in A \\
 & && \sum_{a \in \delta^-(i)} x_a = 1, \forall i \in V_+ \\
 & && \sum_{a \in \delta^-(S)} x_a \geq \left\lceil \frac{d(S)}{C} \right\rceil, \forall S \subseteq V_+ \\
 & && x_a \in \{0,1\}, \forall a \in A \\
 & && \lambda_t \geq 0, \quad t \in T
 \end{aligned}$$

Limitation

- Cuts from the literature over the x variables other than CCs had minimal effect.

BCP over General Extended Formulations

Question

How to find new effective cuts on problems where BCP is being used ?

Some Observations on BCP

Suppose that the pricing is being solved by dynamic programming. We can introduce up to one variable to each transition between states.

1. Cuts over those variables do not change the pricing subproblem.
2. The size of the LPs that are actually solved does not change too.
3. The large number of new variables may allow complex cuts to be expressed in a simple way.
4. The new variables may be very useful to represent odd constraints and/or objective func.

Some Observations on BCP

If you are already solving the pricing by dynamic programming, extending the formulation to include the transition variables is a “free lunch”.

Examples

- On CVRP we can have binary variables x_{ij}^d meaning that a vehicle arrives in j from i with a load of d units.
- Similar capacity-indexed variables for the CMST.
- On VRPTW we can have variables indexed by both capacity and time.

Examples

- On scheduling problems several pseudo-polynomial time-indexed formulations are known.
- On network design with distance constraints (hops in the simpler case), it is possible to use distance-indexed vars.

BCP over Capacity-Indexed Formulations

The Capacity-Indexed Formulation (CIF) for the CMST Gouveia (1995)

$$\begin{aligned}
 \min \quad & \sum_{a \in A} c_a \sum_{d=1}^C x_a^d \\
 s.t. \quad & \sum_{a \in \delta^-(i)} \sum_{d=1}^C x_a^d = 1, \forall i \in V_+ \\
 & \sum_{a \in \delta^-(i)} \sum_{d=1}^C d \cdot x_a^d - \sum_{a \in \delta^+(i)} \sum_{d=1}^{C-1} d \cdot x_a^d = d(i), \forall i \in V_+ \\
 & x_a^d \in \{0,1\}, \forall a \in A, d = 1 \dots C
 \end{aligned}$$

The CIF for the CMST

- mC variables, but only $2n$ constraints,
- Its linear relaxation is weak, equivalent to a single-flow formulation.

The Fixed Charge Network Flow Problem

$$\begin{aligned} \min \quad & \sum_{a \in A} c_a x_a + \sum_{a \in A} f_a w_a \\ \text{s.t.} \quad & \sum_{a \in \delta^-(i)} x_a - \sum_{a \in \delta^+(i)} x_a = d(i), \forall i \in V \\ & 0 \leq x_a \leq u_a w_a, \forall a \in A \\ & w \text{ binary} \end{aligned}$$

Suppose d and u integral

FCNF Capacity-Indexed Reformulation

$$\begin{aligned} \min \quad & \sum_{a \in A} \sum_{d=1}^{u_a} (c_a d + f_a) x_a^d \\ s.t. \quad & \sum_{a \in \delta^-(i)} \sum_{d=1}^{u_a} d \cdot x_a^d - \sum_{a \in \delta^+(i)} \sum_{d=1}^{u_a} d \cdot x_a^d = d(i) \quad , \forall i \in V \\ & \sum_{d=1}^{u_a} x_a^d \leq 1 \quad , \forall a \in A \\ & x \text{ binary} \end{aligned}$$

FCNF Capacity-Indexed Reformulation

- Same value of LP relaxation.
- The CIFs for the CMST and concentrator location are special cases.
- Models several other problems, including many VRP and facility location variants, lot sizing, Steiner, BPP, etc.
- Directly useful for small capacities (say, up to 10).

The Base Equalities for CI-FCNF

- For each vertex i in V :

$$\sum_{a \in \delta^-(i)} \sum_{d=1}^{u_a} d.x_a^d - \sum_{a \in \delta^+(i)} \sum_{d=1}^{u_a} d.x_a^d = d(i)$$

Summing over a set S in V :

$$\sum_{a \in \delta^-(S)} \sum_{d=1}^{u_a} d.x_a^d - \sum_{a \in \delta^+(S)} \sum_{d=1}^{u_a} d.x_a^d = d(S)$$

Extended Capacity Cuts (ECCs)

- An ECC over a set S is any inequality valid for

$$\left\{ \begin{array}{l} \sum_{a \in \delta^-(S)} \sum_{d=1}^{u_a} d \cdot x_a^d - \sum_{a \in \delta^+(S)} \sum_{d=1}^{u_a} d \cdot x_a^d = d(S) \\ \sum_{d=1}^{u_a} x_a^d \leq 1 \quad \forall a \in (\delta^-(S) \cup \delta^+(S)) \\ x \text{ binary} \end{array} \right\}$$

Extended Capacity Cuts (ECCs)

Many known cuts for the previously mentioned problems can be shown to be equivalent or dominated by ECCs.

CMST Capacity Cuts are ECCs

- Pick first equation, relax to \geq , divide by C :

$$\sum_{a \in \delta^-(S)} \sum_{d=1}^C (d/C) \cdot x_a^d - \sum_{a \in \delta^+(S)} \sum_{d=1}^{C-1} (d/C) \cdot x_a^d \geq d(S)/C$$

- Perform simple integral rounding

$$\sum_{a \in \delta^-(S)} \sum_{d=1}^C x_a^d \geq \lceil d(S)/C \rceil$$

Note that the original variables $x_a = \sum_{d=1}^C x_a^d$

Homogeneous ECCs

- Given a set S , define

$$y_d = \sum_{a \in \delta^-(S)} x_a^d \quad \forall d = 1 \dots \max(u_a)$$

$$z_d = \sum_{a \in \delta^+(S)} x_a^d \quad \forall d = 1 \dots \max(u_a)$$

$$D = d(S)$$

- HECCs are cuts over the above y and z integer variables.

HECCs obtained by MIR

The Mixed Integer Rounding (with a suitable sub-additive function) of

$$\sum_{d=1}^C rd.y_d - \sum_{d=1}^{C-1} rd.z_d \leq rD$$

where $r = a/b$, a and b in the range $1 \dots C$, already gives a reasonable family of cuts.

Obtaining better HECCs

When C is small, one can compute the facets of

$$P(C, D) = \left\{ \begin{array}{l} \sum_{d=1}^C d \cdot y_d - \sum_{d=1}^{C-1} d \cdot z_d = D, \\ \sum_{d=1}^C y_d \leq |S|, \\ (y, z) \in Z_+^{2C-1} \end{array} \right\}$$

Facet HECCs

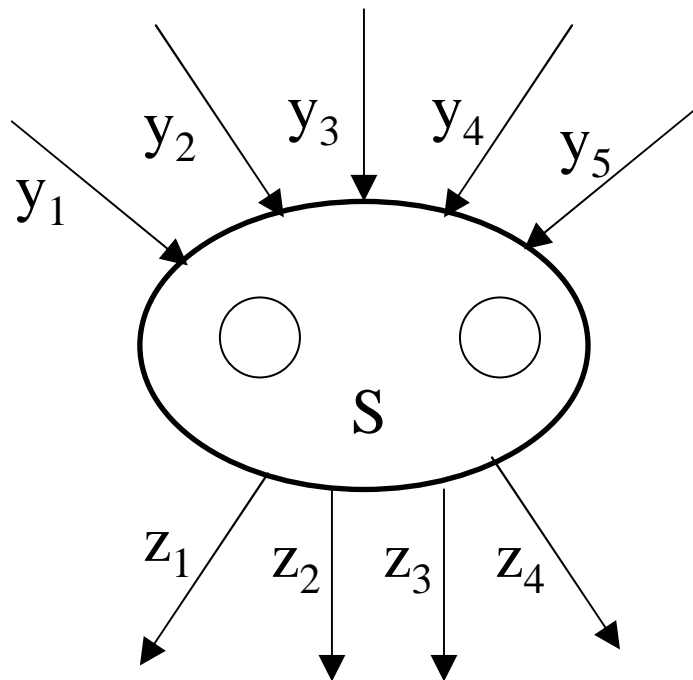
When C is small, polyhedra $P(C,D)$ can be described by few facets. The value of D (we tested up to 10) has little effect on the number of facets.

Examples:

- $P(5,D)$ s have 4 to 6 non-trivial facets.
- $P(10,D)$ s have about 60 non-trivial facets.

$$P(5,2) =$$

$$\left\{ \begin{array}{l} y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 - z_1 - 2z_2 - 3z_3 - 4z_4 = 2, \\ y_1 + y_2 + y_3 + y_4 + y_5 \leq 2, \end{array} \quad (y, z) \in Z_+^9 \right\}$$



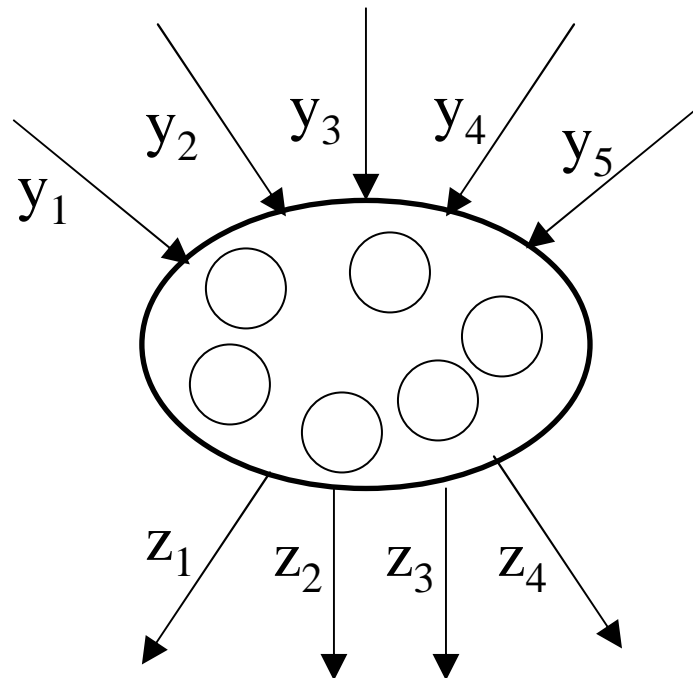
$$\mathbf{C=5, d(S)=2}$$

$$P(5,2) =$$

$$\left\{ \begin{array}{l} y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 - z_1 - 2z_2 - 3z_3 - 4z_4 = 2, \\ y_1 + y_2 + y_3 + y_4 + y_5 \leq 2, \\ y_1 + 2y_2 + 2y_3 + 3y_4 + 4y_5 - z_2 - 2z_3 - 2z_4 \geq 2 \text{ (r = 2/3)}, \\ y_1 + 2y_2 + 2y_3 + 3y_4 + 3y_5 - z_2 - z_3 - 2z_4 \geq 2 \text{ (r = 3/5)}, \\ y_1 + 2y_2 + 2y_3 + 2y_4 + 3y_5 - z_3 - 2z_4 \geq 2, \\ y_1 + 2y_2 + 2y_3 + 2y_4 + 2y_5 - z_4 \geq 2, \\ y_1 + 2y_2 + 2y_3 + 3y_4 + 4y_5 - z_1 - 2z_2 - 2z_3 - 3z_4 \leq 2, \\ y_1 + 2y_2 + 2y_3 + 2y_4 + 3y_5 - z_1 - 2z_2 - 2z_3 - 2z_4 \leq 2, \\ (y, z) \geq 0 \end{array} \right\}$$

$$P(5,6) =$$

$$\left\{ \begin{array}{l} y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 - z_1 - 2z_2 - 3z_3 - 4z_4 = 6, \\ y_1 + y_2 + y_3 + y_4 + y_5 \leq 6, \end{array} \quad (y, z) \in Z_+^9 \right\}$$



$$\mathbf{C=5, d(S)=6}$$

$$P(5,6)=$$

$$\left\{ \begin{array}{l} y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 - z_1 - 2z_2 - 3z_3 - 4z_4 = 6, \\ y_1 + y_2 + y_3 + y_4 + y_5 \leq 6, \\ y_1 + y_2 + y_3 + y_4 + y_5 \geq 2 \quad (r=1/5, \text{ a CC}), \\ y_1 + 2y_2 + 2y_3 + 2y_4 + 3y_5 - z_3 - 2z_4 \geq 4, \\ y_1 + 2y_2 + 2y_3 + 2y_4 + 3y_5 - z_2 - 2z_3 - 2z_4 \leq 4, \\ y_1 - y_5 + z_1 + z_2 + z_3 + 2z_4 \geq 0, \\ (y, z) \geq 0 \end{array} \right\}$$

Better ECCs for large C

It is expected that better cuts can be obtained by exploring the ECC binary knapsack (actually subset-sum) with GUBs structure.

- Lifted covers ?
- Future research: how those cuts can be compared with lifted flow covers on the original formulation ?

BCP with ECCs

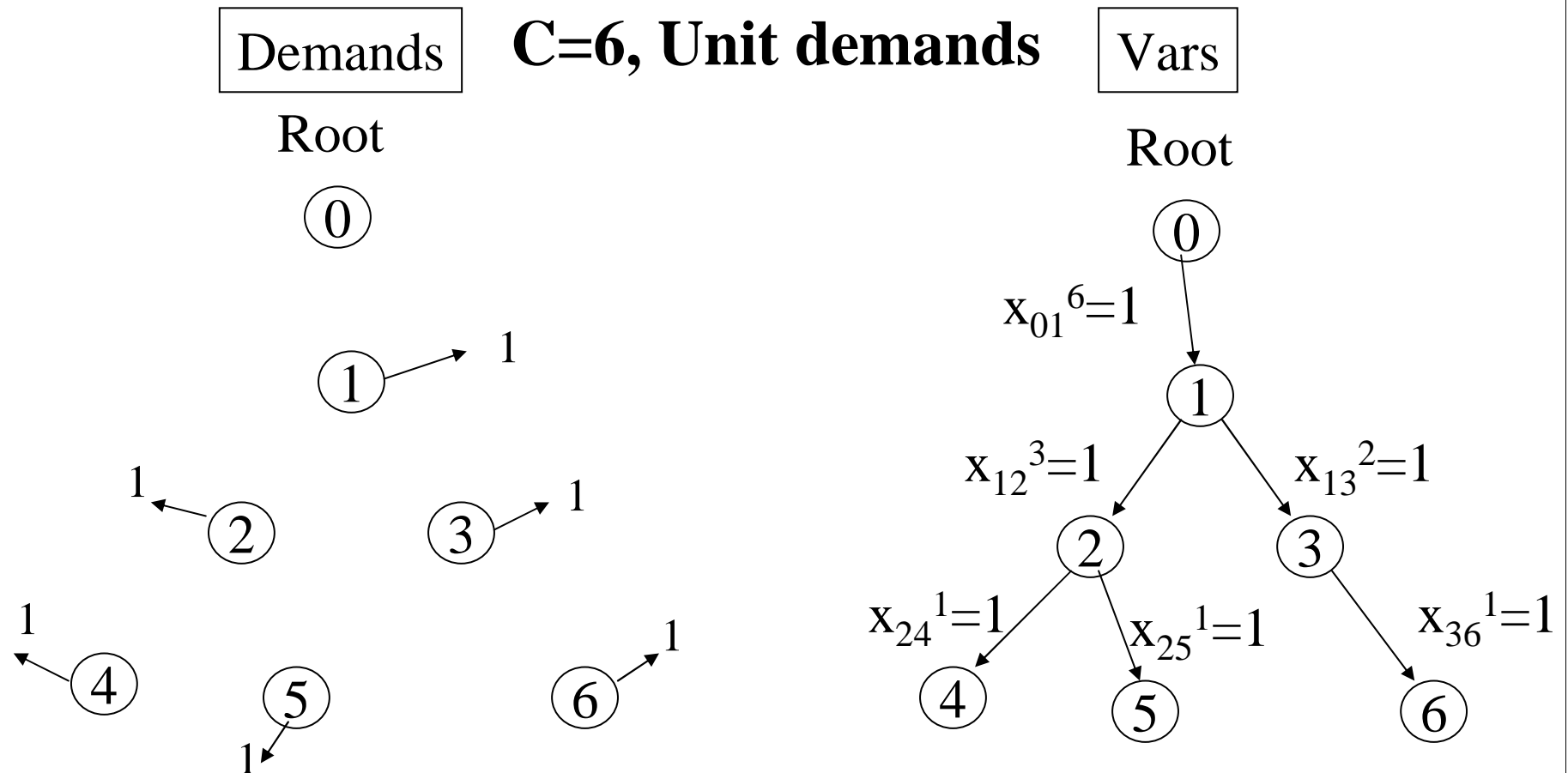
- BCP allows the use of CI formulations even when the capacities are large.
- Moreover, ECCs have shown to produce a nice synergetic effect.

CMST

Uchoa, Fukasawa, Lysgaard, Pessoa,
Poggi de Aragão, Andrade (2005).

- BCP using ECCs solves all instances from the literature with up to 100 vertices, even for large values of C .

CMST : Decomposing a q-arb into capacity-indexed variables



Bounds and times on formerly open instance cm50-r1 C=200

Best BC from the literature	: 1039.9, 3s
RBCP using traditional cuts	: 1093.4, 65s
RBCP using extended cuts	: 1097.5, 148s

Opt: **1098**, 153s = (2 nodes)

Pentium 2.8 GHz.

Bounds and times on formerly open instance cm100-r1 C=200

Best BC from the literature	:	465.9, 11s
RBCP using traditional cuts	:	501.8, 341s
RBCP using extended cuts	:	506.7, 1480s

Opt: **509**, 17227s = (64 nodes)

CMST

The MIR HECCs used are generic CI-FCNF cuts, they know nothing about CMST !

BCP robusto sobre formulações extendidas

Resultados muito fortes já obtidos em alguns outros problemas clássicos.

Muito trabalho a ser feito nessa linha de pesquisa, há várias questões teóricas ainda em aberto. Tb há problemas práticos a serem melhor tratados, incluindo questões de convergência e instabilidade numérica.

Agradecimentos

- Agradeço os organizadores do ELAVIO'07 pela oportunidade de ministrar este mini-curso e ao programa de pós-graduação em Engenharia de produção da UFF pelo apoio a minha pesquisa.
- Agradeço aos doutorandos Haroldo Santos e Lorenza Leão pela ajuda na preparação destes slides.

Propaganda

- O departamento de Engenharia de Produção da UFF recentemente abriu seu programa de doutorado e busca alunos de tempo integral para trabalhar em PI, especialmente em algoritmos de Branch-Cut-and-Price.
- Possibilidade de bolsas de estudos.
- Colaboração ativa com outros grupos de pesquisa no Brasil e no exterior.