Algoritmos Combinatórios: Backtracking

Lucia Moura
lucia@site.uottawa.ca

UFSC, Fevereiro, 2010

Problema da Mochila

Problema da Mochila (Otimização)

Instância: Lucros $p_0, p_1, \ldots, p_{n-1}$
Pesos $w_0, w_1, \ldots, w_{n-1}$
Capacidade da mochila M

Encontre: uma n-tupla $[x_0, x_1, \ldots, x_{n-1}] \in \{0,1\}^n$
tal que o lucro total $P = \sum_{i=0}^{n-1} p_i x_i$ seja maximizado,
sujeito a restrição de capacidade: $\sum_{i=0}^{n-1} w_i x_i \leq M$.

Exemplo

Objetos:	1	2	3	4
pesos (Kg)	8	1	5	4
lucros	\$500	\$1,000	\$ 300	\$ 210

Capacidade da mochila: $M = 10$ Kg.

Duas soluções viáveis e seus lucros:

x_1	x_2	x_3	x_4	lucro
1	1	0	0	\$ 1,500
0	1	1	1	\$ 1,510

Este problema é NP-difícil.

Algoritmo Ingênuo de Backtracking para a "Mochila"

Examine todas as 2^n tuplas e guarde uma de lucro máximo.

Variaveis Globais: $X, OptP, OptX$.

Algoritmo $\textsc{Mochila1}\ (l)$

if $(l = n)$ then
  if $\sum_{i=0}^{n-1} w_i x_i \leq M$ then $CurP \leftarrow \sum_{i=0}^{n-1} p_i x_i$;
    if $(CurP > OptP)$ then
      $OptP \leftarrow CurP$;
      $OptX \leftarrow [x_0, x_1, \ldots, x_{n-1}]$;
else $x_l \leftarrow 1$; $\textsc{Mochila1}\ (l+1)$;
  $x_l \leftarrow 0$; $\textsc{Mochila1}\ (l+1)$;

Chamada principal: $OptP \leftarrow -1$; $\textsc{Mochila1}\ (0)$.

Tempo: 2^n n-tuplas com tempo $\Theta(n)$ cada uma. O tempo total é $\Theta(n2^n)$.

Note: nem todas as n-tuplas são viáveis, mas o algoritmo testa todas (toda a árvore de busca é examinada). Nós melhoraremos este algoritmo!!!

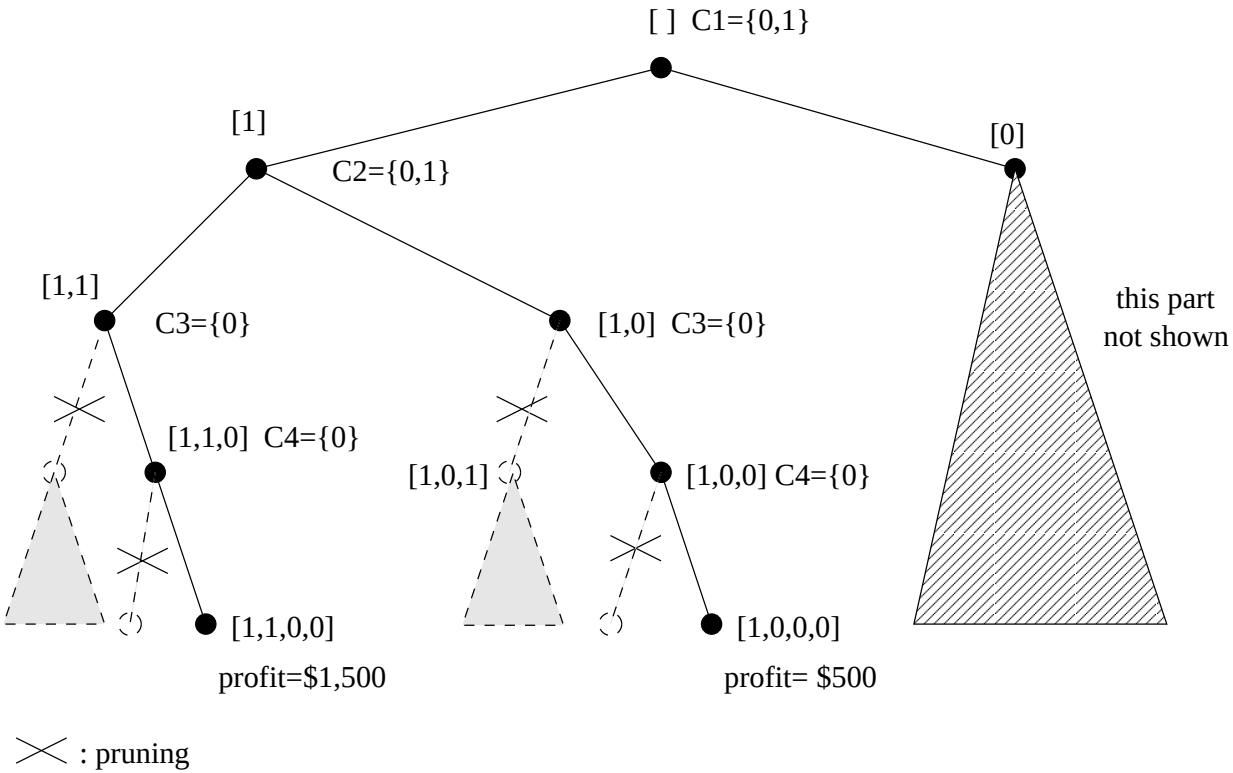
Um Algoritmo Geral de Backtracking

- Represente uma solução como uma lista: $X = [x_0, x_1, x_2, \ldots]$.
- Cada $x_i \in P_i$ (conjunto de possibilidades)
- Dada uma solução parcial: $X = [x_0, x_1, \ldots, x_{l-1}]$, podemos usar restrições do problema para limitar as opções para x_l a um conjunto $\mathcal{C}_l \subseteq P_l$ (conjunto de opções).
- Ao computar $\mathcal{C}_l$, nós podamos a árvore de busca, já que para todo $y \in P_l \setminus \mathcal{C}_l$ a subárvore com raiz $[x_0, x_1, \ldots, x_{l-1}, y]$ não é considerada.

Parte da árvore de busca para o exemplo anterior da mochila:

w_i	8	1	5	4
p_i	\$500	\$1,000	\$ 300	\$ 210

$M = 10.$



Algoritmo Geral de Backtracking com Podas

Variaveis Globais $X = [x_0, x_1, \ldots]$, $\mathcal{C}_l$, for $l = 0, 1, \ldots$

Algoritmo $\textsc{Backtrack}\ (l)$

if ($X = [x_0, x_1, \ldots, x_{l-1}]$ e' uma solucao viavel) then

Processe X

Compute $\mathcal{C}_l$;

for each $x \in \mathcal{C}_l$ do

$x_l \leftarrow x$;

$\textsc{Backtrack}(l+1)$;

Backtracking com podas para o Problema da Mochila

Variaveis Globais $X, OptP, OptX$.

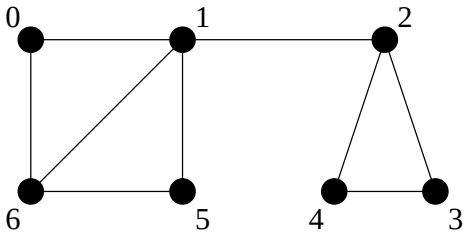
Algoritmo $\textsc{Mochila2}\ (l, CurW)$

if $(l = n)$ then if $(\sum_{i=0}^{n-1} p_i x_i > OptP)$ then
$OptP \leftarrow \sum_{i=0}^{n-1} p_i x_i$;
$OptX \leftarrow [x_0, x_1, \ldots, x_{n-1}]$;
if $(l = n)$ then $\mathcal{C}_l \leftarrow \emptyset$
else if $(CurW + w_l \leq M)$ then $\mathcal{C}_l \leftarrow \{0, 1\}$;
else $\mathcal{C}_l \leftarrow \{0\}$;
for each $x \in \mathcal{C}_l$ do
$x_l \leftarrow x$;
$\textsc{Mochila2}\ (l + 1, CurW + w_l x_l)$;

Chamada principal: $\textsc{Mochila2}\ (0, 0)$.

Backtracking: Gerar todos os cliques

PROBLEMA: Todos os cliques
INSTÂNCIA: um grafo $G = (V, E)$.
GERE: todos os cliques de G sem repetição



Cliques (e cliques maximais): $\emptyset, \{0\}, \{1\}, \ldots, \{6\}$,
$\{0,1\}, \{0,6\}, \{1,2\}, \{1,5\}, \{1,6\}, \{2,3\}, \{2,4\}, \{3,4\}, \{5,6\}$,
$\{0,1,6\}, \{1,5,6\}, \{2,3,4\}$.

Definition

Clique em $G(V, E)$: $C \subseteq V$ tal que para todo $x, y \in C$, $x \neq y$, $\{x, y\} \in E$.
Clique maximal: um clique que não esteja propriamente contido em um outro clique.

Muitos problemas combinatórios podem ser reduzidos a encontrar cliques (ou cliques de tamanho máximo):

- O maior conjunto independente (de vértices) em G: é o mesmo que o maior clique no grafo complemento $\overline{G}$.
- Cobertura exata de um conjunto por subconjuntos: encontre um clique com propriedades especiais.
- Encontre um sistema de triplas de Steiner de ordem v: encontre o maior clique num grafo especial.
- Encontre todos os sistemas de sub-conjuntos intersectantes: encontre todos os cliques em um grafo especial.
- Etc.

Num algoritmo de backtracking, $X = [x_0, x_1, \ldots, x_{l-1}]$ é uma solução parcial $\iff$ $\{x_0, x_1, \ldots, x_{l-1}\}$ é um clique.
Mas não queremos gerar $k!$ vezes o mesmo k-clique.
$[0, 1]$ pode ser extendido a $[0, 1, 6]$
$[0, 6]$ pode ser extendido a $[0, 6, 1]$

Então estabelecemos que as soluções parciais devem estar ordenadas:
$x_0 < x_1 < x_2 < \ldots < x_{l-1}$.

Seja $S_{l-1} = \{x_0, x_1, \ldots, x_{l-1}\}$, para $X = [x_0, x_1, \ldots, x_{l-1}]$.
O **conjunto de opções** é:
se $l = 0$ então $\mathcal{C}_0 = V$
se $l > 0$ então

$$\begin{aligned} \mathcal{C}_l &= \{v \in V \setminus S_{l-1} : v > x_{l-1} \text{ and } \{v, x\} \in E \text{ for all } x \in S_{l-1}\} \\ &= \{v \in \mathcal{C}_{l-1} \setminus \{x_{l-1}\} : \{v, x_{l-1}\} \in E \text{ and } v > x_{l-1}\} \end{aligned}$$

Então,
$\mathcal{C}_0 = V$
$\mathcal{C}_l = \{v \in \mathcal{C}_{l-1} \setminus \{x_{l-1}\} : \{v, x_{l-1}\} \in E \text{ and } v > x_{l-1}\}$, for $l > 0$

Para computar $\mathcal{C}_l$, defina:
$A_v = \{u \in V : \{u, v\} \in E\}$ (vértices adjacentes a v)
$B_v = \{v+1, v+2, \ldots, n-1\}$ (vértices de índice maior que v)
$\mathcal{C}_l = A_{x_{l-1}} \cap B_{x_{l-1}} \cap \mathcal{C}_{l-1}$.

Para **detectar se um clique é maximal** (com respeito a inclusão de conjuntos):
Calcule N_l, o conjunto de vértices que podem extender S_{l-1}:
$N_0 = V$
$N_l = N_{l-1} \cap A_{x_{l-1}}$.
S_{l-1} é maximal $\iff$ $N_l = \emptyset$.

Algoritmo $\textsc{TodosCliques}(l)$
Global: X, $\mathcal{C}_l (l = 0, \ldots, n-1)$, pre-computados: A_x, B_x, $\forall x$ vertice.

if $(l = 0)$ then output $([\,])$;
else output $([x_0, x_1, \ldots, x_{l-1}])$;
if $(l = 0)$ then $N_l \leftarrow V$;
else $N_l \leftarrow A_{x_{l-1}} \cap N_{l-1}$;
if $(N_l = \emptyset)$ then output ("maximal");
if $(l = 0)$ then $\mathcal{C}_l \leftarrow V$;
else $\mathcal{C}_l \leftarrow A_{x_{l-1}} \cap B_{x_{l-1}} \cap \mathcal{C}_{l-1}$;
for each $(x \in \mathcal{C}_l)$ do
    $x_l \leftarrow x$;
    $\textsc{TodosCliques}(l+1)$;

Chamada principal: $\textsc{TodosCliques}(0)$.

Análise do Caso Médio de TodosCliques

Seja $\mathcal{G}_n$ o conjunto de todos os grafos com n vértices.
$|\mathcal{G}_n| = 2^{\binom{n}{2}}$ (bijeção entre $\mathcal{G}_n$ e todos os subconjuntos do conjunto de 2-subconjuntos de $\{1, 2, \ldots, n\}$).
Suponha que os grafos em $\mathcal{G}_n$ são equiprováveis como entrada para o algoritmo (isto é, suponha uma distribuição uniforme em $\mathcal{G}_n$).

Seja $T(n)$ o tempo médio usado pelo algoritmo TodosCliques para grafos em $\mathcal{G}_n$, e seja $\overline{c}(n)$ o número médio de cliques em um grafo em $\mathcal{G}_n$.
Então, $T(n) \in O(n\overline{c}(n))$.

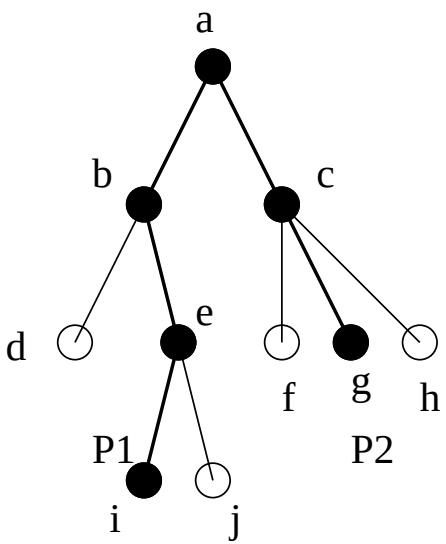
É possível provar que

$$\overline{c}(n) \leq (n+1)n^{\log_2 n}, \text{ para } n \geq 4.$$

Então, $T(n) \in O(n^{\log_2 n + 2})$.

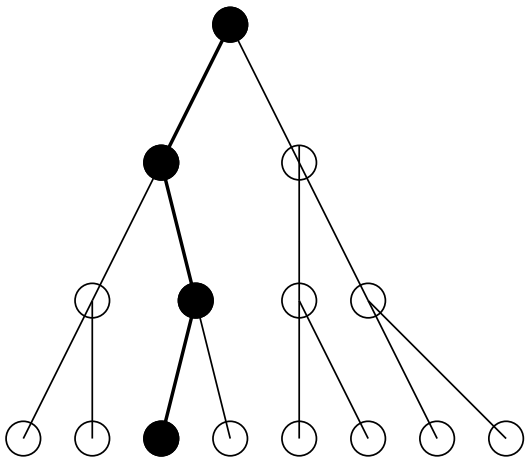
Estimando o tamanho da árvore de backtracking

Árvore do Espaço de Estados do Algoritmo de Backtracking:
tamanho (número de vértices) da árvore $= 10$

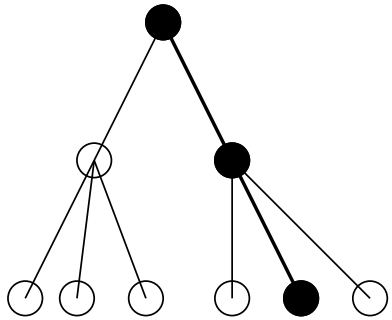


Escolhemos aleatoriamente caminhos iniciando na raiz e terminando em uma folha da árvore, para estimar o número de vértices da árvore.

Escolhendo caminho:	P_1	P_2
Tamanho estimado da árvore:	$N(P_1) = 15$	$N(P_2) = 9$



P1



P2

Escolhendo caminho:	P_1	P_2
Tamanho estimado da árvore:	$N(P_1) = 15$	$N(P_2) = 9$

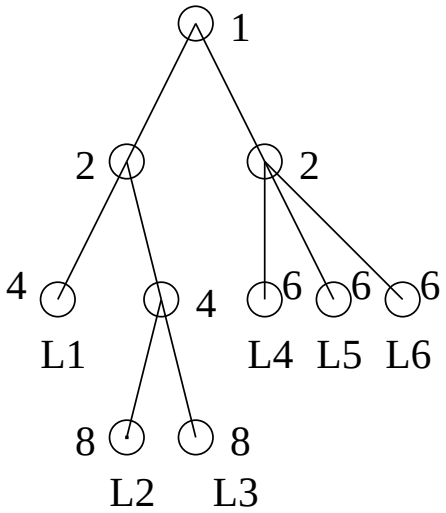
Escolha de um caminho aleatório:
A cada vértice da árvore, escolha um filho uniformemente ao acaso.

Para cada folha L, seja $P(L)$ a probabilidade que L seja atingida por este processo.
Seja $\overline{N}$ o valor esperado de $N(L)$, isto é:

$$\overline{N} = \sum_{L\text{folha}} P(L)N(L)$$

Provaremos que $\overline{N}$ é exatamente o número de vértices da árvore.

No exemplo anterior, em cada vértice de T anote o número estimado de vértices de mesmo nível, se o caminho escolhido passasse por este vértice.



$P(L_1) = 1/4,\ P(L_2) = P(L_3) = 1/8,\ P(L_4) = P(L_5) = P(L_6) = 1/6$
$N(L_1) = 1 + 2 + 4 = 7 \quad N(L_2) = N(L_3) = 1 + 2 + 4 + 8 = 15$
$N(L_4) = N(L_5) = N(L_6) = 1 + 2 + 6 = 9$

$$\overline{N} = \sum_{i=1}^{6} P(L_i)N(L_i) = \frac{1}{4} \times 7 + 2 \times (\frac{1}{8} \times 15) + 3 \times (\frac{1}{6} \times 9) = 10 = |T|$$

Na prática, para **estimar** $\overline{N}$, escolha aleatoriamente k caminhos chegando a folhas $L_1, L_2, \ldots, L_k$, e calcule a média de $N(L_i)$:

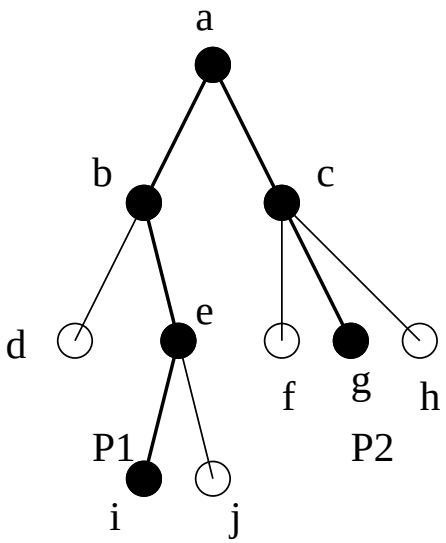
$$N_{est} = \frac{\sum_{i=1}^{k} N(L_i)}{k}$$

Ou seja, calculamos a média dos valores retornados em k execuções de:

Algorithm EstimarTamanhoBacktrack()

$s \leftarrow 1$; $N \leftarrow 1$; $l \leftarrow 0$;
Compute $\mathcal{C}_0$;
while $(\mathcal{C}_l \neq \emptyset)$ do
  $c \leftarrow |\mathcal{C}_l|$;
  $s \leftarrow c * s$;
  $N \leftarrow N + s$;
  $x_l \leftarrow$ um elemento aleatorio de $\mathcal{C}_l$;
  Compute $\mathcal{C}_{l+1}$ para $[x_0, x_1, \ldots, x_l]$;
  $l \leftarrow l + 1$;
return N;

No exemplo abaixo, escolhendo aleatoriamente k=2 caminhos.



P_1:	l	C_l	c	x_l	s	N
					1	1
	0	b, c	2	b	2	3
	1	d, e	2	e	4	7
	2	i, j	2	i	8	<u>15</u>
	3	$\emptyset$				

P_1:	l	C_l	c	x_l	s	N
					1	1
	0	b, c	2	c	2	3
	1	f, g, h	3	g	6	<u>9</u>
	2	$\emptyset$				

O tamanho estimato da árvore é $N_{est} = \frac{15+9}{2} = 12$.

Teorema

Para uma árvore de espaço de estados T, seja P o caminho escolhido pelo algoritmo EstimarTamanhoBacktrack.
Se $N = N(P)$ é o valor retornado pelo algoritmo, então o valor esperado de N é $|T|$.

Prova.

Defina a seguinte função nos vértices de T:

$$S([x_0, x_1, \ldots, x_{l-1}]) = \begin{cases} 1, & \text{se } l = 0 \\ |\mathcal{C}_{l-1}| \times S([x_0, x_1, \ldots, x_{l-2}]), & \text{caso contrario} \end{cases}$$

($s \leftarrow c * s$ no algoritmo)

O algoritmo computa: $N(P) = \sum_{Y \in P} S(Y)$.

$P = P(X)$ é um caminho em T da raiz até a folha $X = [x_0, x_1, \ldots, x_{l-1}]$.
Seja $X_i = [x_0, x_1, \ldots, x_i]$.
A probabilidade que $P(X)$ seja escolhido é:

$$\frac{1}{|\mathcal{C}_0(x_0)|} \times \frac{1}{|\mathcal{C}_1(x_1)|} \times \ldots \times \frac{1}{|\mathcal{C}_{l-1}(x_{l-1})|} = \frac{1}{S(X)}.$$

Então,

$$\begin{aligned}
\overline{N} &= \sum_{X \in \mathcal{L}(T)} prob(P(X)) \times N(P(X)) \\
&= \sum_{X \in \mathcal{L}(T)} \frac{1}{S(X)} \sum_{Y \in P(X)} S(Y) \\
&= \sum_{Y \in T} \sum_{\{X \in \mathcal{L}(T) : Y \in P(X)\}} \frac{S(Y)}{S(X)} \\
&= \sum_{Y \in T} S(Y) \sum_{\{X \in \mathcal{L}(T) : Y \in P(X)\}} \frac{1}{S(X)}
\end{aligned}$$

Provaremos a seguir o fato:

$$\sum_{\{X \in \mathcal{L}(T) : Y \in P(X)\}} \frac{1}{S(X)} = \frac{1}{S(Y)}.$$

Seja Y um vértice não folha . Se Z é um filho de Y e Y tem c filhos, então $S(Z) = c \times S(Y)$.
Então,

$$\sum_{\{Z : Z \text{ is a child of } Y\}} \frac{1}{S(Z)} = c \times \frac{1}{c \times S(Y)} = \frac{1}{S(Y)}$$

Iterando esta equação até todos os Z's serem folhas:

$$\frac{1}{S(Y)} = \sum_{\{X : X \text{ e' uma folha descendente de } Y\}} \frac{1}{S(X)}$$

Então o fato está provado!

Então,

$$\begin{aligned}
\overline{N} &= \sum_{Y \in T} S(Y) \sum_{\{X \in \mathcal{L}(T) : Y \in P(X)\}} \frac{1}{S(X)} \\
&= \sum_{Y \in T} S(Y) \frac{1}{S(Y)} \\
&= \sum_{Y \in T} 1 = |T|.
\end{aligned}$$

O teorema está provado.