CI1238 - Otimização

Aula 12 - Backtracking (Parte 2)

Professor Murilo V. G. da Silva

Departamento de Informática

Universidade Federal do Paraná

2026 / Primeiro Semestre

Backtracking para Problema do Caixeiro Viajante

Seja G um grafo completo ponderado com pesos positivos nas arestas.

- Podemos assumir $V(G) = \{0, 1, ..., n-1\}$
- Solução para CAIXEIRO é circuito hamiltoniado de custo mínimo em G

Reduzindo um pouco espaço de busca: as soluções viáveis são restritas as tuplas:

- $X = [x_0, x_1, \cdots, x_{n-1}]$ tal que $x_0 = 0$
- e.g., o circuito hamiltoniano $[2, 5, 1, 0, 3, 4, 6]$ corresponde às tuplas:
  - $[0, 3, 4, 6, 2, 5, 1]$ e $[0, 1, 5, 2, 6, 4, 3]$

Solução parcial viável: Tupla $[x_0, .., x_l]$, $l \leq n-1$
($x_0 = 0$, tupla pode ser estendida para solução viável)

- Note que apenas folhas da árvore contém soluções
- Escolhas viáveis:
  $C_0 = 0$,
  $C_l = C_{l-1} \setminus \{x_{l-1}\}$

Backtracking para Problema do Caixeiro Viajante

Algorithm 4.10: TSP1 (ℓ)

global $C_\ell \quad (\ell = 0, 1, \ldots, n-1)$

if $\ell = n$

then $\begin{cases} C \leftarrow \text{cost}([x_0, \ldots, x_{n-1}]) \\ \textbf{if } C < OptC \\ \quad \textbf{then} \begin{cases} OptC \leftarrow C \\ OptX \leftarrow [x_0, \ldots, x_{n-1}] \end{cases} \end{cases}$

if $\ell = 0$

then $C_\ell \leftarrow \{0\}$

else $\begin{cases} \textbf{if } \ell = 1 \\ \quad \textbf{then } C_\ell \leftarrow \{1, \ldots, n-1\} \\ \quad \textbf{else } C_\ell \leftarrow C_{\ell-1} \setminus \{x_{\ell-1}\} \end{cases}$

for each $x \in C_\ell$

do $\begin{cases} x_\ell \leftarrow x \\ \text{TSP1}(\ell + 1) \end{cases}$

Algoritmo Genérico de Backtracking

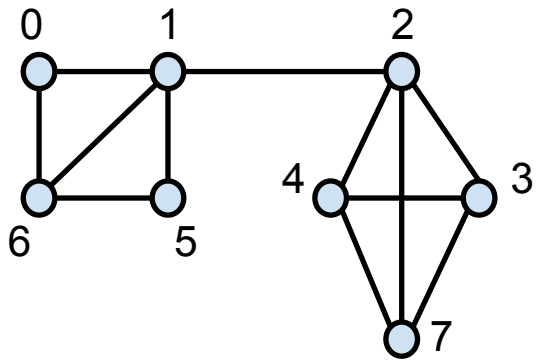
Relembrando:

Algorithm 4.2: BACKTRACK (ℓ)

global $X, \mathcal{C}_\ell \quad (\ell = 0, 1, \ldots)$
comment: $X = [x_0, x_1, \ldots]$
if $[x_0, x_1, \ldots, x_{\ell-1}]$ is a feasible solution
  then process it
Compute $\mathcal{C}_\ell$
for each $x \in \mathcal{C}_\ell$
  do $\begin{cases} x_\ell \leftarrow x \\ \text{BACKTRACK}(\ell + 1) \end{cases}$

Backtracking para problema da clique

Dado um $G = (V, E)$, um conjunto de vértices que induz um subgrafo completo de G é chamado de uma clique de G.



- Exemplo: $\{0, 1, 6\}$ é uma clique de G
  $\{0, 1, 6, 5\}$ não é uma clique de G

Cliques do grafo do exemplo acima:

- Tamanho 0: $\{\}$ (na definição do livro, existem cliques de tamanho zero)
- Tamanho 1: $\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$
- Tamanho 2: $\{0, 1\}, \{0, 6\}, \{1, 2\}, \{1, 5\}, \{1, 6\}, \{2, 4\}, \{2, 3\}, \{2, 7\}, \{3, 4\}, \{3, 7\}, \{4, 7\}, \{5, 6\}$
- Tamanho 3: $\{0, 1, 6\}, \{1, 5, 6\}, \{2, 3, 4\}, \{2, 3, 7\}, \{2, 4, 7\}$ $\{3, 4, 7\}$
- Tamanho 4: $\{2, 3, 4, 7\}$

Cliques maximais: $\{1, 2\}, \{0, 1, 6\}, \{1, 5, 6\}, \{2, 3, 4, 7\}$ Clique máxima: $\{2, 3, 4, 7\}$

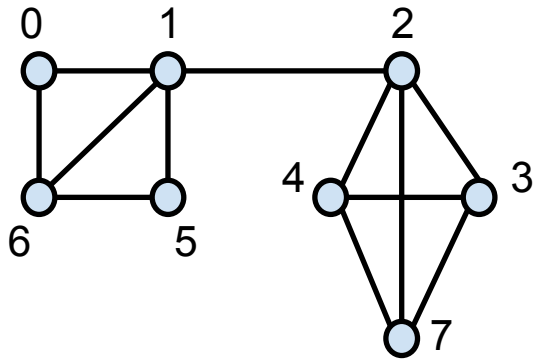
Usando backtracking para enumerar todas as cliques

Vamos começar com um algoritmo de backtracking o seguinte problema:
Dado um grafo G de entrada:

- Listar todas as suas cliques
- Identificar entre estas quais cliques são maximais
  - Este algoritmo pode ser usado como base problemas de busca ou otimização envolvendo cliques em grafos

    (e.g., encontrar a clique máxima)

Considere o grafo abaixo:



Neste exemplo o algoritmo irá:

- listar todas as suas 28 cliques (ver slide anterior)
- identificar que $\{1,2\}$, $\{0,1,6\}$, $\{1,5,6\}$, $\{2,3,4,7\}$ são as cliques maximais

Enumerando todas as cliques

Descrevendo o Algoritmo de *backtracking* passo a passo:

- Espaço de busca: $[x_0, ..., x_k]$, $0 \leq k \leq n$
  x_i é um vértice do grafo $G = (V, E)$
  i.e., $P_0 = V(G)$, $P_1 = V(G)$, $P_2 = V(G)$...

  $\Rightarrow$ todas da k-tuplas estão no espaço de busca

  Backtracking: ao invés de expandir todos ramos da árvore,
  a idéia é restringir cada x_i apenas aos vértices disponíveis em C_i

- Solução parcial $[x_0, ..., x_{l-1}]$ o conjunto C_l:
  - Solução parcial: $[x_0, ..., x_{l-1}]$ é solução parcial se $\{x_0, ..., x_{l-1}\}$ é clique
    A ideia é começar com a tupla vazia [ ] e ir construindo recursivamente outras soluções parciais.
  - Computando conjunto de escolhas C_l evitando repetição de vértices:
    - Base: $C_0 = V$ (no passo 0 todo vértice é uma escolha válida)
    - Dado $S_{l-1} = \{x_0, ..., x_{l-1}\}$ (vértices da clique solução parcial)
      $C_l = \{v \in V \setminus S_{l-1} \mid vx \in E(G), \text{ para cada } x \in S_{l-1}\}$.

Enumerando todas as cliques

Do slide anterior:

- $C_l = \{v \in V(G) \setminus S_{l-1} \mid vx \in E(G), \text{ para cada } x \in S_{l-1}\}$

Será conveniente reescrever C_l da seguinte maneira:

$$C_l = \{v \in C_{l-1} \setminus \{x_{l-1}\} \mid vx_{l-1} \in E(G)\}$$

- Escolhas disponíveis para o l-ésimo vértice: as mesmas escolhas disponíveis no passo anterior, exceto x_{l-1}
- O vértice x_{l-1} escolhido no passo anterior está agora na solução parcial, portanto não estará mais disponível em C_l.

Assim não geramos tuplas com vértices repetidos, mas ainda temos um problema:

Cada clique de tamanho k será gerada $k!$ vezes
Estamos gerando todas as k-permutações

- Exemplo: a clique da solução $[x_1, x_2, x_3]$ é a mesma clique das soluções $[x_1, x_3, x_2]$, $[x_2, x_1, x_3]$, $[x_2, x_3, x_1]$, etc...

Enumerando todas as cliques

Resolvendo este problema: Criamos uma ordenação (arbitrária) dos vértices do grafo:

- Estabelecemos uma relação para os vértices tal que $v_1 < v_2 < \cdots < v_n$

Reescrevendo (novamente!) as escolhas:

$$C_l = \{v \in C_{l-1} \mid \{v, x_{l-1}\} \in E \text{ e } v > x_{l-1}\}$$

- A condição $v > x_{l-1}$ garante que cada clique apareça um vez.

  Note: No exemplo da página anterior, a tupla $[x_1, x_2, x_3]$ tem os índices aparecendo em ordem. Nas demais tuplas $[x_1, x_3, x_2]$, $[x_2, x_1, x_3]$, $[x_2, x_3, x_1]$, ... isso não ocorre e portanto não serão consideradas

Enumerando todas as cliques

Do slide anterior, as escolhas para o vértice x_l são dadas pelo conjunto:

$$C_l = \{v \in C_{l-1} \mid vx_{l-1} \in E(G) \text{ e } v > x_{l-1}\}$$

Notação:

- $A_v = \{u \in V(G) \mid uv \in E(G)\}$ (notação no livro para vizinhança de v)
- $B_v = \{u \in V(G) \mid u > v\}$ (vértices com índices maiores que v)
- Note: A_v e B_v podem ser computados em pré-processamento

Reescrevendo C_l pela última vez(!):

$$C_l = A_{x_{l-1}} \cap B_{x_{l-1}} \cap C_{l-1}$$

Última definição: Para cada $X = [x_0, ..., x_{l-1}]$, vamos definir o conjunto N_l.

- $N_0 = V$; $N_l = N_{l-1} \cap A_{x_{l-1}}$
- Fato: X é clique maximal $\Leftrightarrow N_l = \emptyset$.

  (note: A definição recursiva de N_l é exatamente a definição de C_l antes da última otimização que fizemos em C_l em que que a ordem dos vértices é levada em consideração)

Enumerando todas as cliques

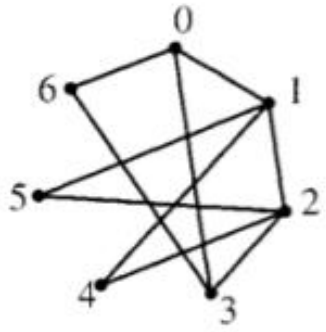
Algorithm 4.4: ALLCLIQUES (ℓ)

global $X, \mathcal{C}_\ell \quad (\ell = 0, 1, \ldots, n-1)$
comment: every X generated by the algorithm is a clique
if $\ell = 0$
  then output ([ ])
  else output ($[x_0, \ldots, x_{\ell-1}]$)
if $\ell = 0$
  then $N_\ell \leftarrow \mathcal{V}$
  else $N_\ell \leftarrow A_{x_{\ell-1}} \cap N_{\ell-1}$
if $N_\ell = \emptyset$
  then $\{x_0, \ldots, x_{\ell-1}\}$ is a maximal clique
if $\ell = 0$
  then $\mathcal{C}_\ell \leftarrow \mathcal{V}$
  else $\mathcal{C}_\ell \leftarrow A_{x_{\ell-1}} \cap B_{x_{\ell-1}} \cap \mathcal{C}_{\ell-1}$
for each $x \in \mathcal{C}_\ell$
  do $\begin{cases} x_\ell \leftarrow x \\ \text{ALLCLIQUES}(\ell + 1) \end{cases}$

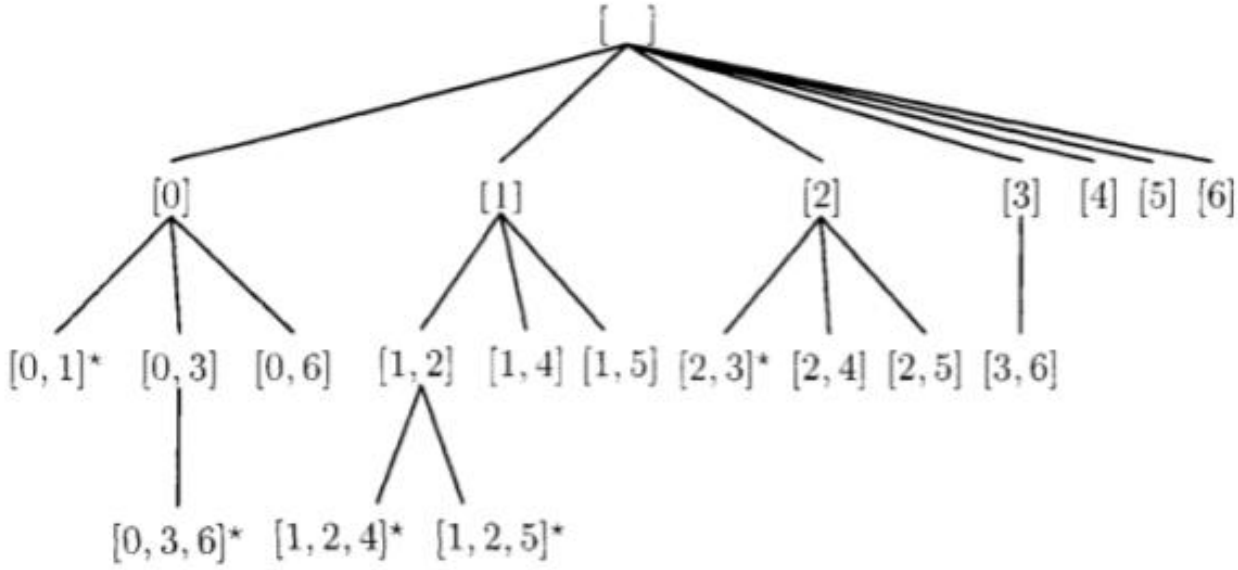
A primeira chamada do algoritmo é feita com $l = 0$.

Exercício: Adapte o Algoritmo para resolver o problema da Clique Máxima.

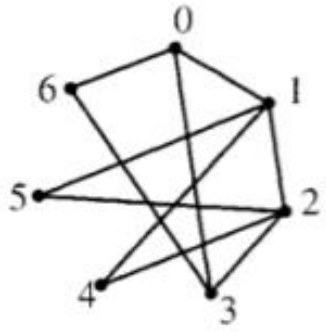
Backtracking para Problema de Listar Cliques



v	A_v	B_v
0	1, 3, 6	1, 2, 3, 4, 5, 6
1	0, 2, 4, 5	2, 3, 4, 5, 6
2	1, 3, 4, 5	3, 4, 5, 6
3	0, 2, 6	4, 5, 6
4	1, 2	5, 6
5	1, 2	6
6	0, 3	



Backtracking para Problema de Listar Cliques



v	A_v	B_v
0	1, 3, 6	1, 2, 3, 4, 5, 6
1	0, 2, 4, 5	2, 3, 4, 5, 6
2	1, 3, 4, 5	3, 4, 5, 6
3	0, 2, 6	4, 5, 6
4	1, 2	5, 6
5	1, 2	6
6	0, 3