BACKTRACKING & BRANCH-AND-BOUND

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

Agenda

1 Introdução

2 Backtracking

3 Branch-and-bound

4 Bibliografia

Introdução

Backtracking e branch-and-bound

- Permite resolver instâncias maiores de problemas combinatoriais
- Melhoramento da busca exaustiva
- No pior caso, precisará explorar todas as possibilidades

Conceito central (explícito ou implícito): árvore de espaço de estados

- Constrói uma solução parcial em busca da solução completa
- A partir desta, explora outras soluções parciais
- Não existindo, retrocede para tentar opções não exploradas

Branch-and-bound: aplicado somente a problemas de otimização

Ordem de geração da árvore

- Backtracking: normalmente em profundidade
- Branch-and-bound: normalmente melhor primeiro

Agenda

1 Introdução

2 Backtracking

3 Branch-and-bound

4 Bibliografia

Problema das n-rainhas

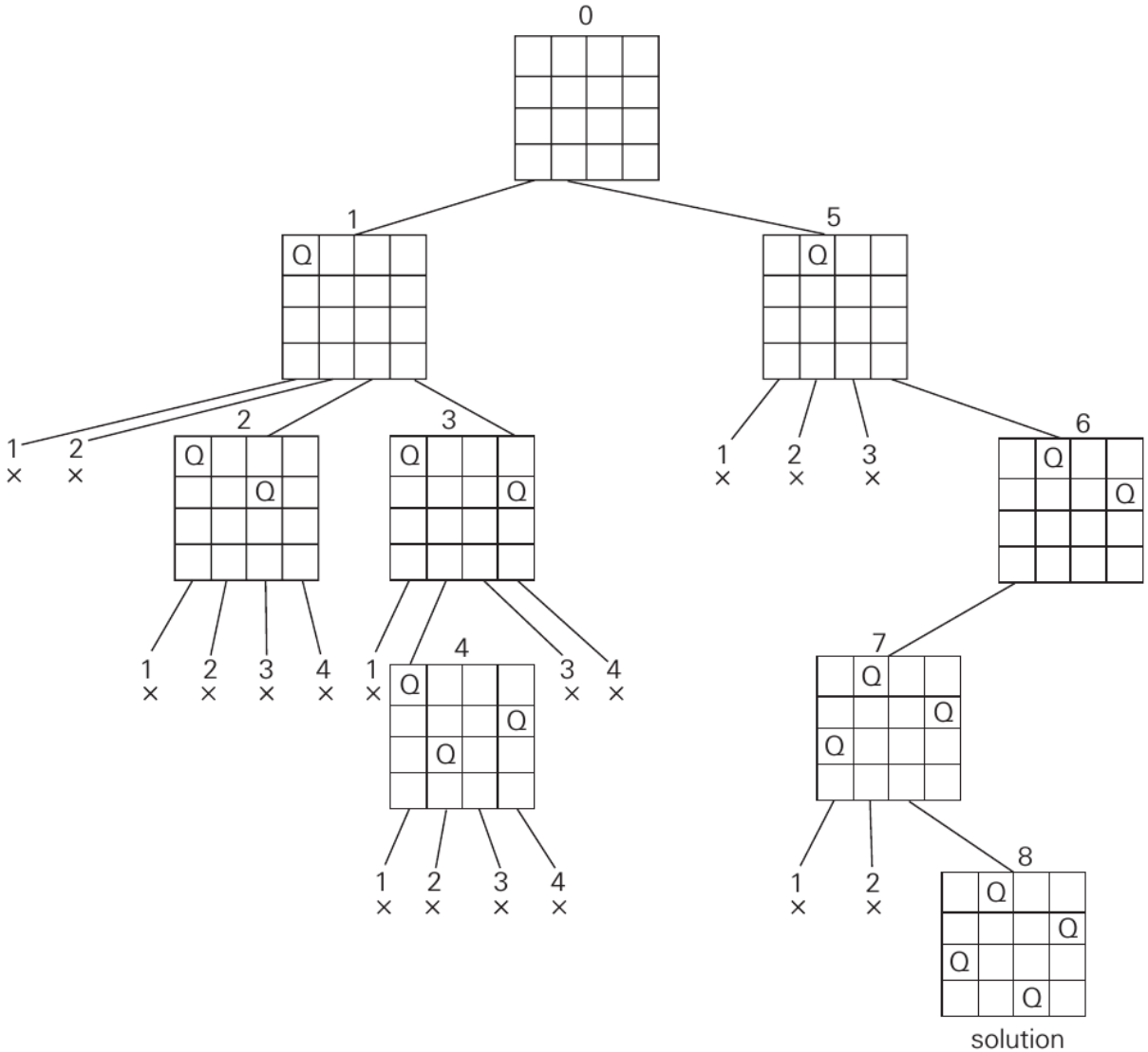
Posicionar n rainhas em um tabuleiro $n \times n$ de forma que não exista nenhuma rainha sob ataque de outra rainha

- $n = 1$: solução trivial
- $n = 2 \vee n = 3$: não existe solução

Problema se resume em escolher a coluna (ou linha) de cada rainha

- Inicialmente: posiciona $1^{\underline{a}}$ rainha na $1^{\underline{a}}$ posição possível
- Tenta posicionar a próxima: não sendo possível, retrocede
- Se conseguir posicionar todas, encontrou uma solução
  - Pode parar, ou pode continuar para encontrar outras

Problema das n-rainhas

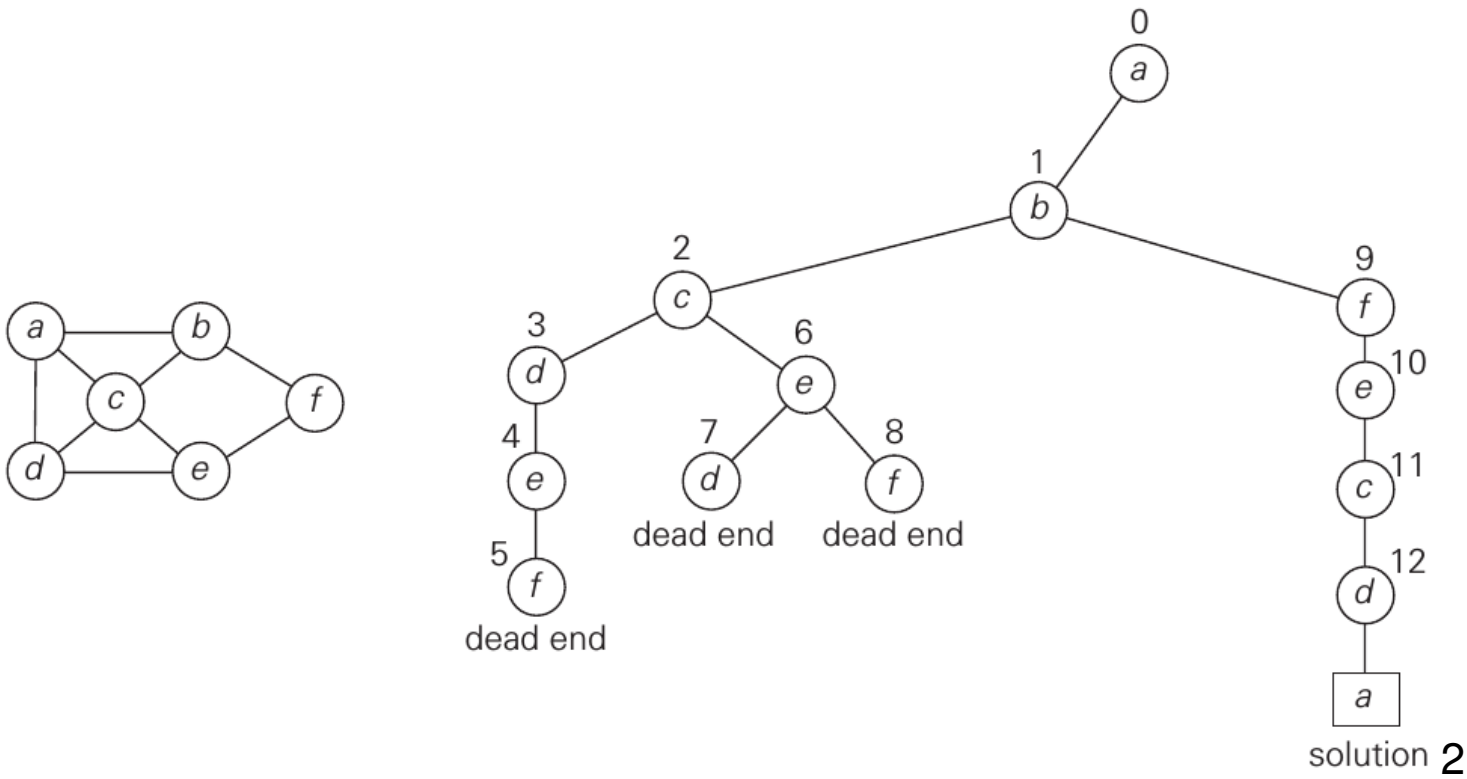


[1]

[1] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Problema do circuito Hamiltoniano

Realizar um circuito simples que passa por todo vértice do grafo



[2] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Problema da soma dos subconjuntos

Seja $A = \{a_1, \cdots, a_n\}$ um conjunto de inteiros, encontrar um $A' \subseteq A$, tal que a soma de seus elementos seja igual a $d \in \mathbb{N}$.

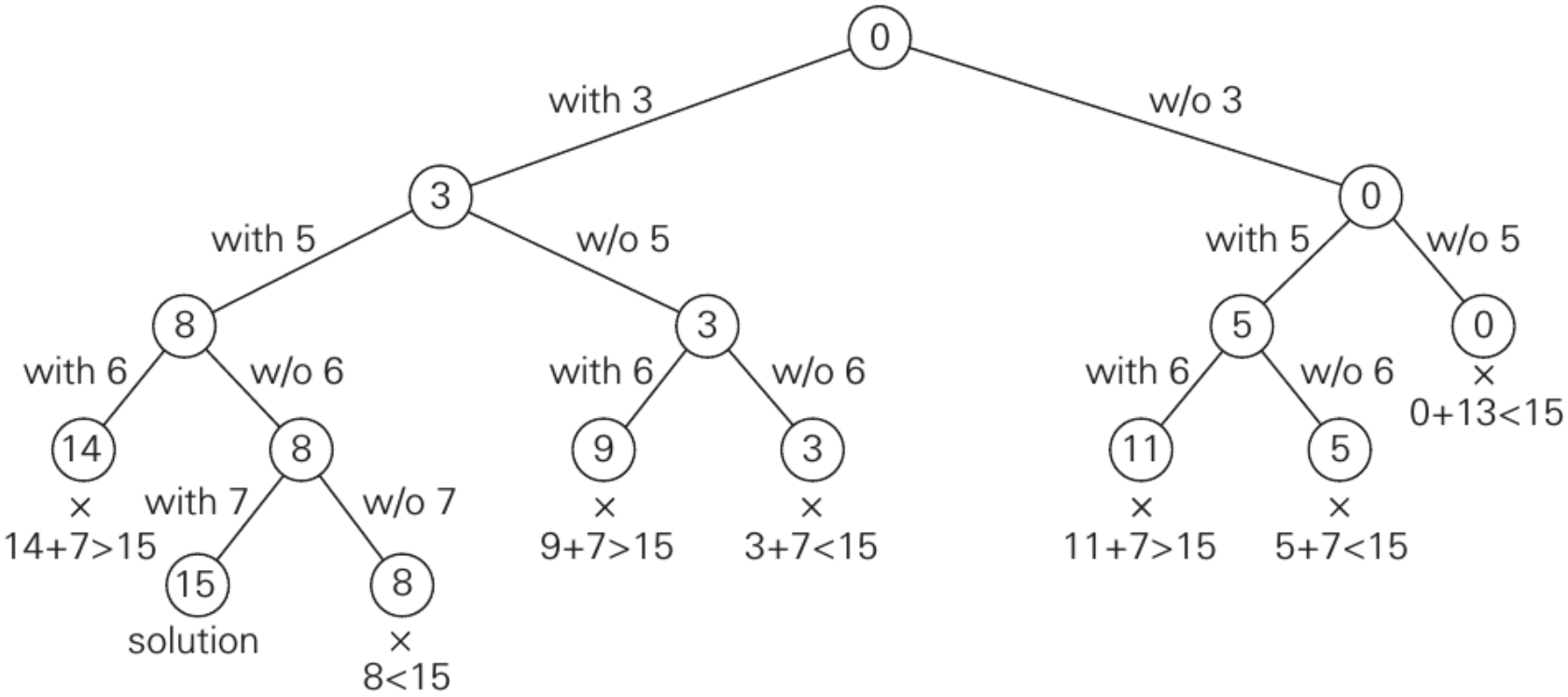
- É conveniente considerar os elementos em ordem crescente $\forall\, i \forall\, j (1 \leq i, j \leq \mid A \mid \wedge\, i < j \Rightarrow a_i < a_j)$

Retroceder se:

- $s + a_{i+1} > d$ (i.e., s é muito grande)
- $(s + \sum_{j=i+1}^{n} a_j) < d$ (i.e., s é muito pequeno)

Exemplo: $A = \{3, 5, 6, 7\}$ e $d = 15$

Problema da soma dos subconjuntos



3

[3] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Backtracking

Em resumo:

- No pior caso, vai ser necessário gerar todas as possibilidades
- Eficiência varia por problema e por instância

Otimizações

- Explorar simetria de problemas combinatoriais
  - n-rainhas: soluções obtidas por reflexão ou rotação
- Reorganizar dados de uma certa instância
  - *subset-sum*: ordenar os valores

Outras aplicações

- Coloração de grafos
- *Knight's tour*

Agenda

1 Introdução

2 Backtracking

3 Branch-and-bound

4 Bibliografia

Branch-and-bound

Técnica para problemas de otimização

- Solução viável: ponto no espaço de busca de um problema que satisfaz as restrições do problema (e.g., circuito Hamiltoniano)
- Solução ótima: solução viável com melhor valor considerando uma função objetivo (e.g., menor circuito Hamiltoniano)

Dois elementos adicionais em relação à backtracking:

- Para cada nó: um limite (bound) para o melhor valor da função objetivo (considerando os descendentes deste nó)
- Conhecer a melhor solução até um dado momento

Expande a partir do melhor limite possível

Se um nó possui um limite pior do que a melhor solução atual, ele é ignorado (pruned / podado)

Problema da atribuição de tarefas

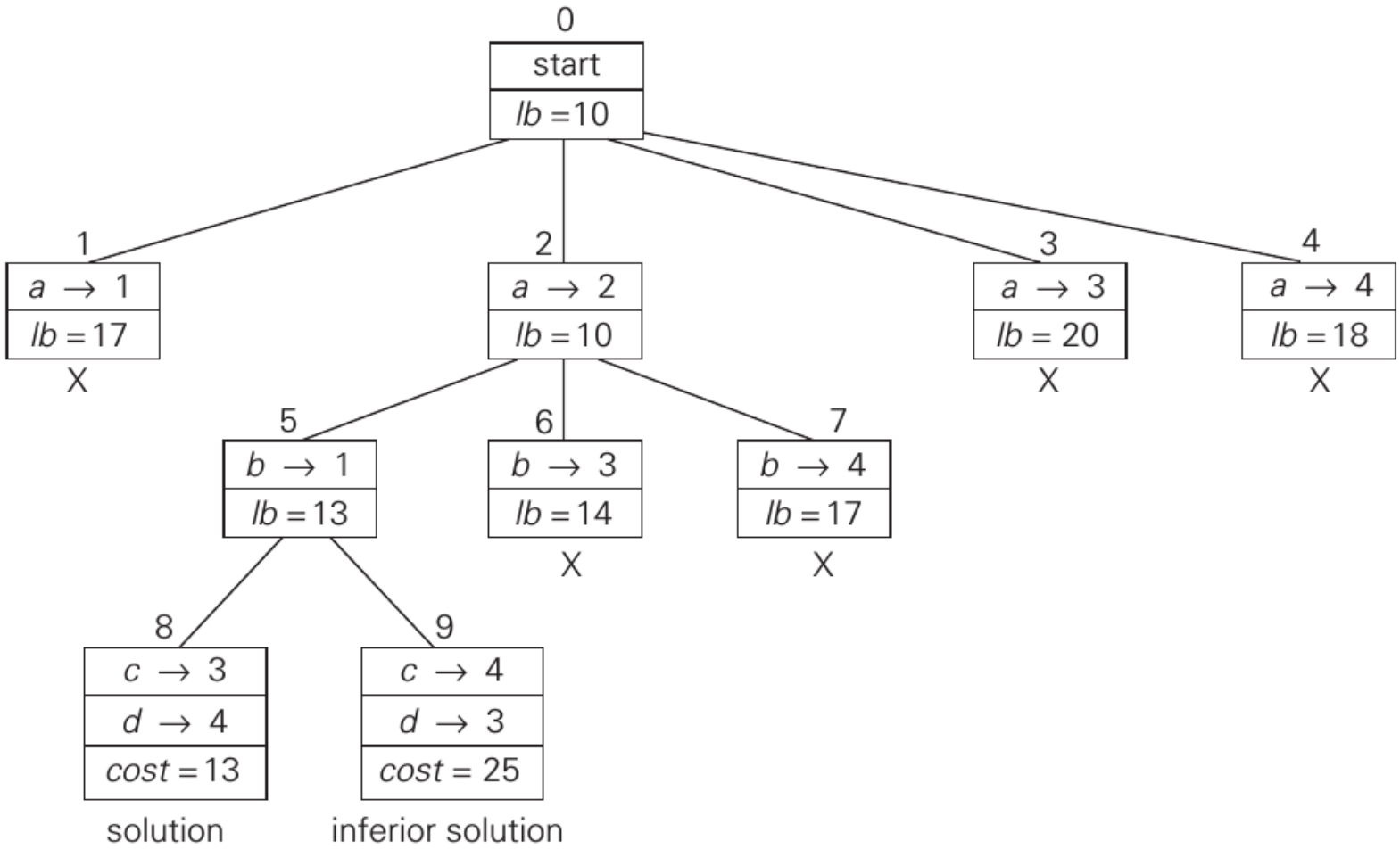
Atribuir *n* pessoas a *n* tarefas minimizando o custo de execução

$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$ [4]

Limite mínimo (lower bound): soma dos menores valores de cada linha

[4] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Problema da atribuição de tarefas



Existe um algoritmo polinomial: Hungarian method

[5] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Problema da mochila

Dado n itens com peso w_i e valor v_i, onde $i = 1, 2, \cdots, n$, e uma capacidade W, encontrar o subconjunto de itens mais valioso que cabe na mochila.

Solução criada a partir da construção de uma árvore binária

- Filho à esquerda: inclusão do elemento
- Filho à direita: exclusão do elemento

Limite superior (upper bound): valor total atual + o produto da capacidade restante pelo item ainda não-selecionado de melhor custo-benefício

- $ub = v + (W - w))(v_{i+1}/w_{i+1})$

Itens ordenados decrescentemente a partir do seu custo-benefício (v_i/w_i)

Problema da mochila

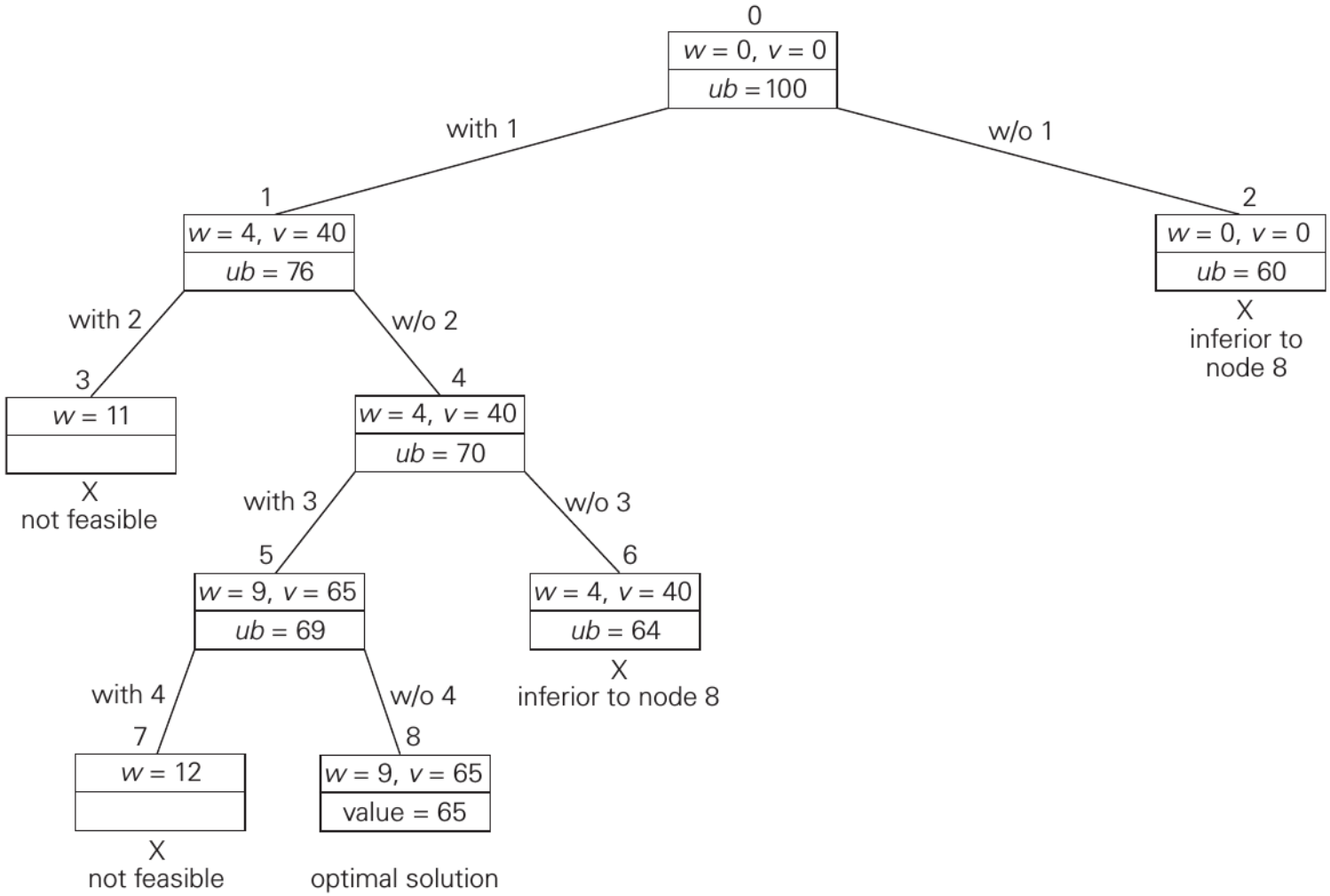
item	weight	value	$\frac{\text{value}}{\text{weight}}$
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4

The knapsack's capacity W is 10.

6

[6] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Problema da mochila



[7] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Problema do caixeiro viajante

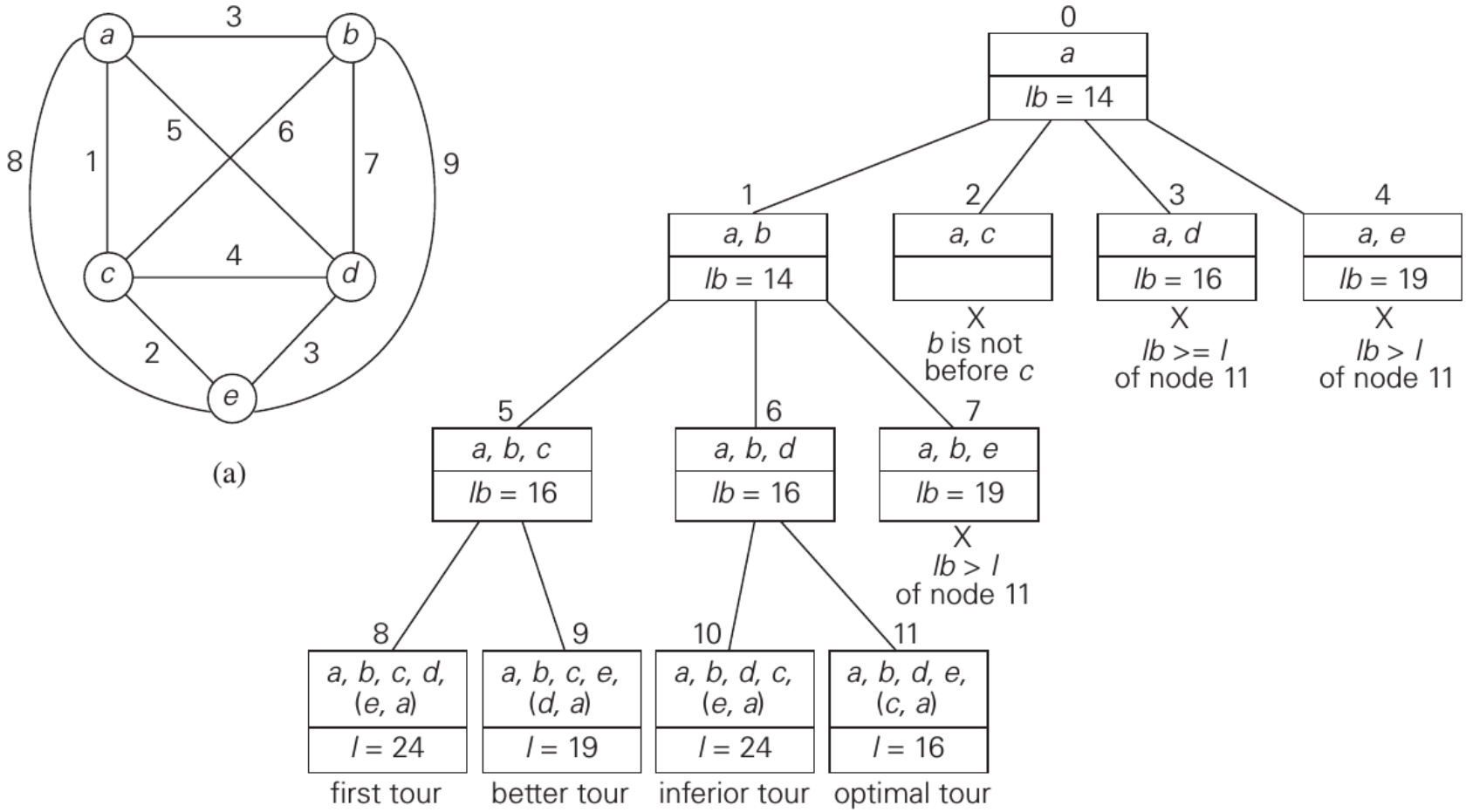
Encontrar o menor circuito Hamiltoniano para um grafo G

Limite inferior (simples): encontrar a menor distância entre duas cidades e multiplicar pelo número de cidades n

Limite inferior (melhor): para cada cidade i, somar a distância de i para as duas cidades mais próximas, por fim, dividir o valor final por 2 (e calcular o teto se todas as distâncias forem inteiras)

- À medida que a árvore é construída, ajustar o cálculo anterior considerando as arestas já escolhidas

Problema do caixeiro viajante



8

[8] Fonte: A. Levitin. Introduction to the Design and Analysis of Algorithms. 2011.

Branch-and-bound

Nem sempre abordagem best-first é a melhor

- IA: outras estratégias para árvores de espaço de estados

Desafio de uma boa função de limite inferior/superior

- Precisa ser fácil de computador
- Não pode ser muito simplista (efeito na poda)

Agenda

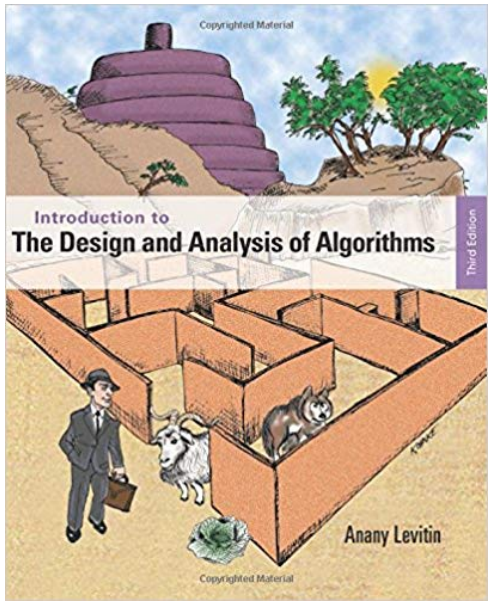
1 Introdução

2 Backtracking

3 Branch-and-bound

4 Bibliografia

Bibliografia + leitura recomendada



Capítulo 12 (pp. 423–440)
Anany Levitin.
Introduction to the Design and Analysis of Algorithms.
3a edição. Pearson. 2011.

BACKTRACKING & BRANCH-AND-BOUND

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil