Técnicas de Projeto (Parte 5)
Projeto e Análise de Algoritmo

Felipe Cunha

Pontifícia Universidade Católica de Minas Gerais

Técnicas de Projeto

1) Branch and Bound

2) Algoritmos Aproximativos

Branch and Bound

- Existem duas abordagens principais para lidar com problemas de complexidade não-polinomial ou intratáveis:
  - Usar uma estratégia que garanta uma solução exata para o problema, mas que não garanta chegar a esta solução em tempo polinomial
  - Usar um algoritmo aproximado, que possa chegar a uma solução não necessariamente exata, mas bem aproximada (sub-ótima) em tempo polinomial

Branch and Bound

- A abordagem exata inclui as técnicas de força-bruta e branch-and-bound
- A idéia de backtracking pode ser usada em problemas de otimização, onde deseja-se minimizar ou maximizar uma função-objetivo, geralmente sujeita a restrições
- Em relação ao algoritmo de backtracking, o algoritmo de branch-and-bound requer dois itens adicionais:
  - Uma maneira de estabelecer, para cada nó da árvore um limite sobre o melhor valor da função-objetivo em qualquer solução que possa ser obtida pela adição de mais componentes à solução parcial representada pelo nó
  - O valor da melhor solução encontrada até o momento.

Branch and Bound

- Se estas informações estiverem disponíveis podemos comparar o valor limite de um nó com o valor da melhor solução até o momento e:
  - Se o valor do limite não for melhor, o nó é podado
  - Senão será expandido

Branch and Bound

- Assim, um caminho de busca é encerrado quando:
  - O valor do limite do nó não é melhor do que o valor da melhor solução até o momento
  - O nó representa uma solução que viola as restrições do problema, ou
  - O subconjunto das soluções possíveis representadas pelo nó consiste em um único ponto

- Ao invés de gerar um único filho do nó mais promissor como no *backtracking*, geramos todos os filhos e expandimos o mais promissor
- Nós mais promissores são os que têm o melhor limite inferior ou superior

Branch and Bound

Exemplo: Problema da mochila 0/1

Deseja-se carregar em uma mochila um subconjunto de objetos que some o maior valor. Cada objeto tem um peso e a mochila tem uma capacidade máxima. Os objetos não podem ser partidos.

Para a mochila de capacidade W=10 e os seguintes objetos:

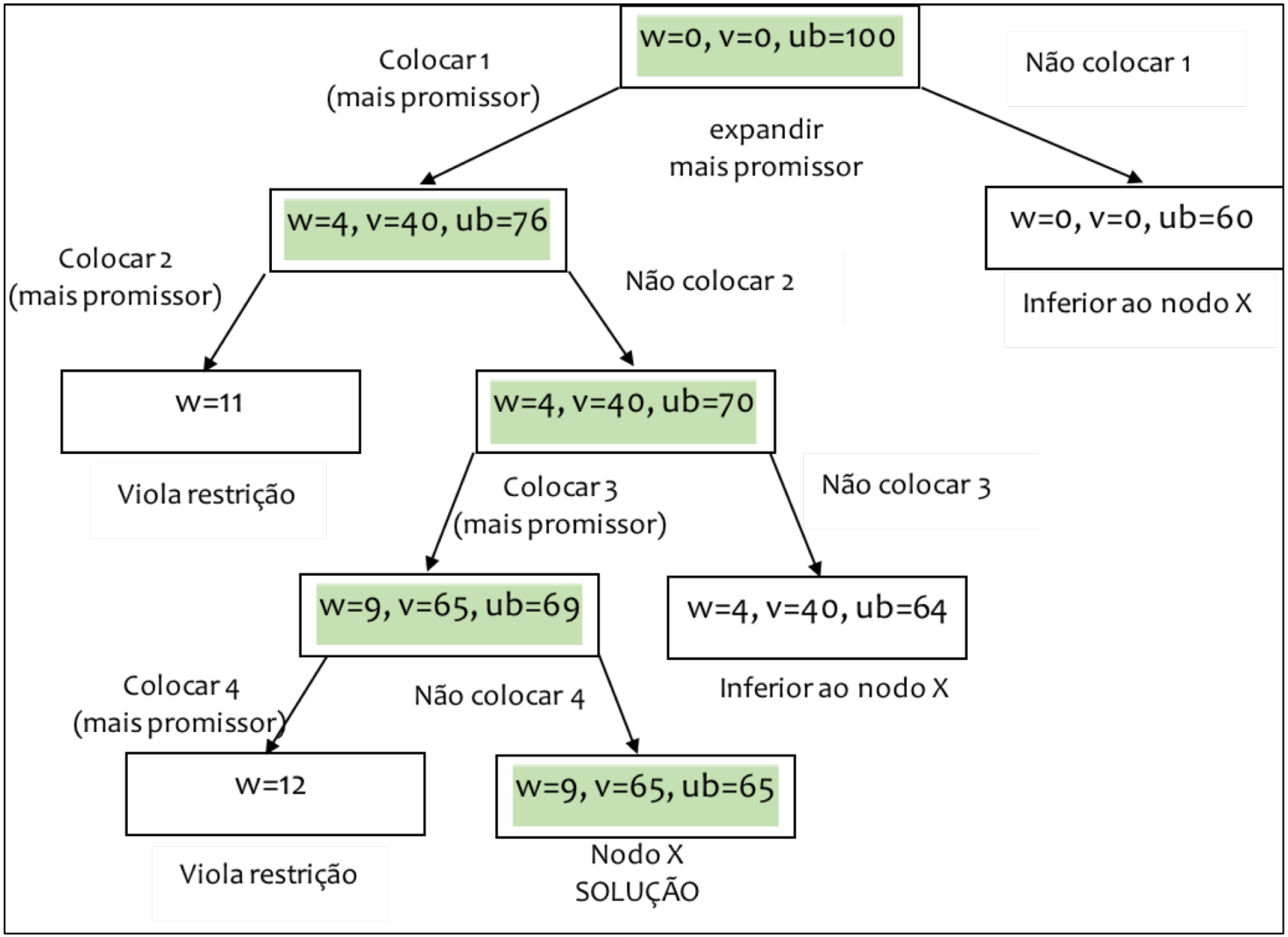
Item (j)	Peso (w)	Valor (v)	Valor/peso
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

Branch and Bound

Para computarmos o limite superior, somaremos *v*, o valor total dos objetos já selecionados, ao produto entre a capacidade restante da mochila e a melhor relação valor/peso entre os objetos restantes:

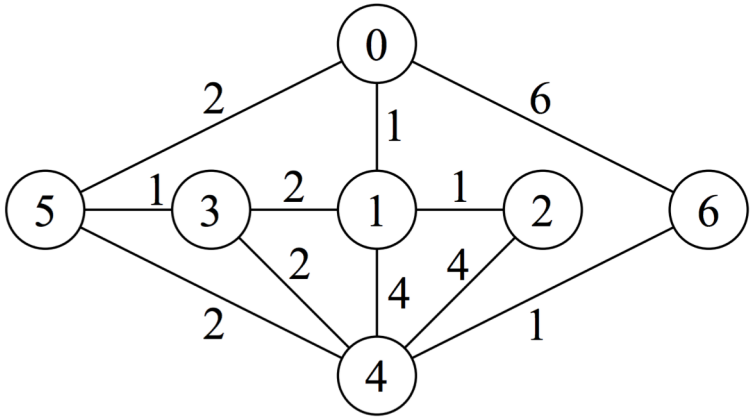
$$ub_{t+1} = \sum_{j=1}^{t} v_j + (W - \sum_{j=1}^{t} w_j) \max_{k \notin S} (v_k / w_k)$$

Branch and Bound



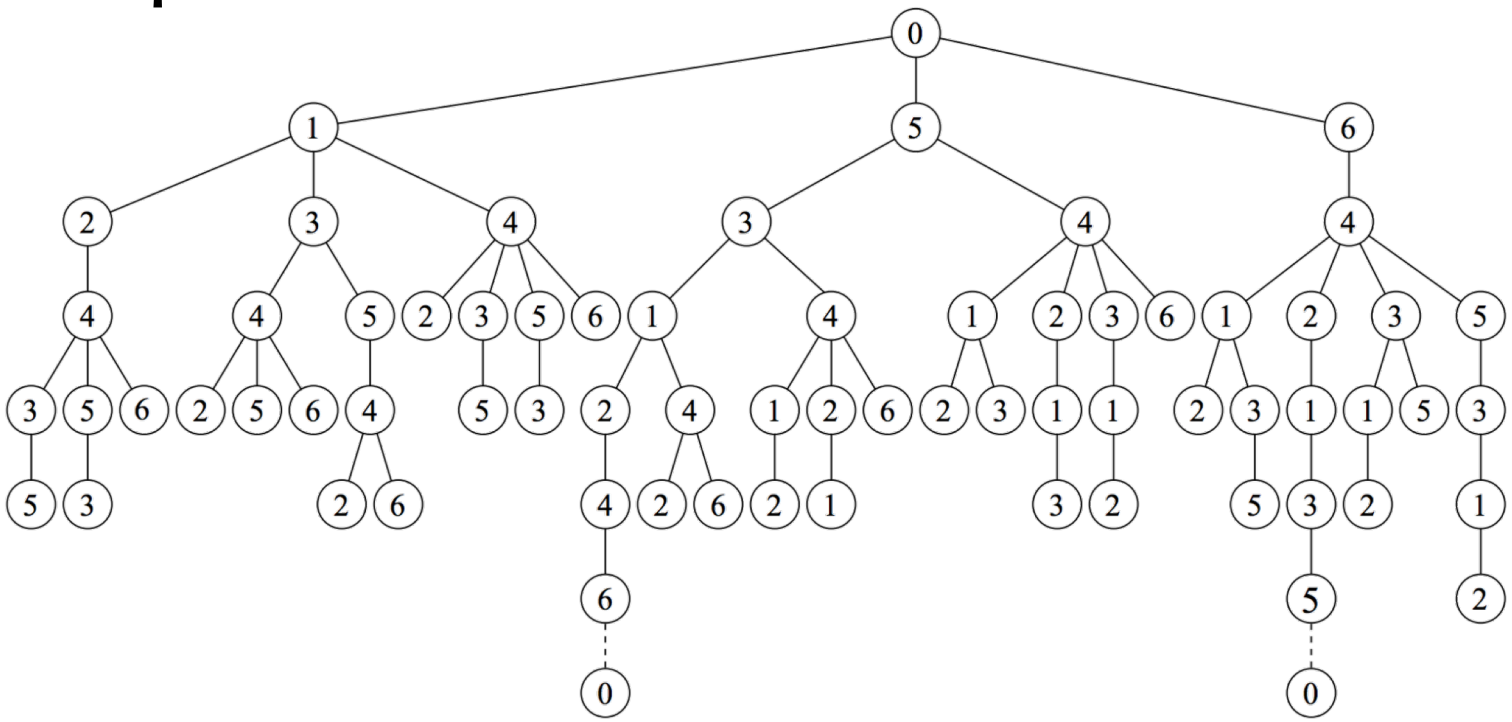
Branch and Bound

- **Exemplo: Ciclo de Hamilton**



Branch and Bound

- **Exemplo: Ciclo de Hamilton**



Branch and Bound

- Outra saída para tentar diminuir o número de chamadas a *Visita( )*
- A ideia é cortar a pesquisa tão logo se saiba que não levará a uma solução
- Corta chamadas a *Visita( )* tão logo se chegue a um custo para qualquer caminho que seja maior que um caminho solução já obtido
- Neste caso, podemos evitar chamadas a *Visita( )* **se o custo do caminho corrente for maior ou igual ao melhor caminho obtido até o momento**

Algoritmos Aproximados

- Problemas de otimização que estão na classe NP não possuem soluções gerais eficientes
- Algoritmos aproximados podem ser usados para se encontrarem soluções próximas do ótimo, em tempo polinomial
- Estes algoritmos possuem diferentes níveis de sofisticação, entretanto, a maioria deles constitui-se de algoritmos gulosos baseados em alguma heurística
- Heurísticas são regras de bom senso, baseadas na experiência e na forma como resolvemos problemas naturalmente

Algoritmos Aproximados

Problema do Caixeiro Viajante: visitar n cidades, passando por cada uma única vez, saindo e chegando de uma mesma cidade. Todas as cidades estão interligadas e deseja-se a solução de menor custo.

- Heurística do vizinho mais próximo
- Heurística do vizinho mais distante

Algoritmos Aproximados

Exemplo: Problema do Empacotamento (Bin Packing): Utilizar o menor número possível de caixas para empacotar *n* objetos de tamanhos variados.

- Heurística do maior primeiro
- Heurística do melhor primeiro

Algoritmos Aproximados

Exemplo: Problema cobertura de vértices

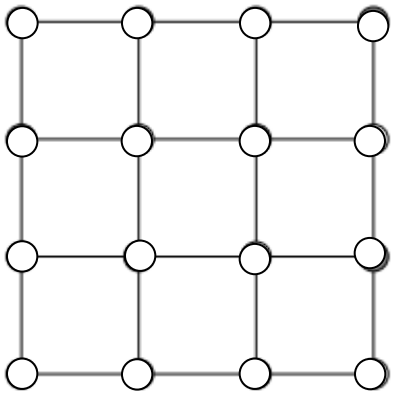
Input: Grafo G = (V,E) não-dirigido

Output: **Menor** V' $\subseteq$ V cobrindo todas as arestas de E (qualquer aresta {u,v} tem uma de suas extremidades em V')

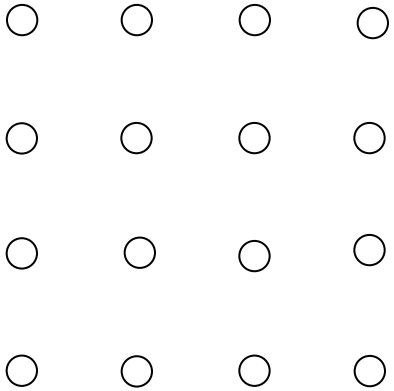
Algoritmos Aproximados

- Constrói um **casamento máximo**
  - casamento = conjunto de arestas que não possuem vértices em comum
  - Casamento máximo = se acrescentar mais uma aresta deixa de ser casamento
  - Construção do casamento máximo: **feito em tempo polinomial**

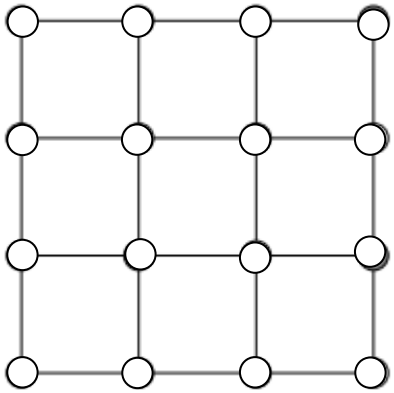
Algoritmos Aproximados



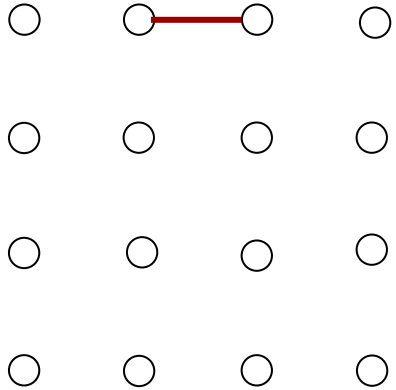
Grafo G



Algoritmos Aproximados

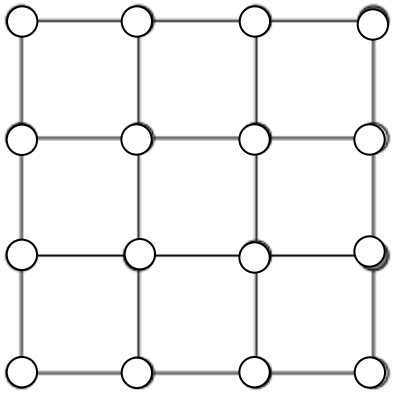


Grafo G

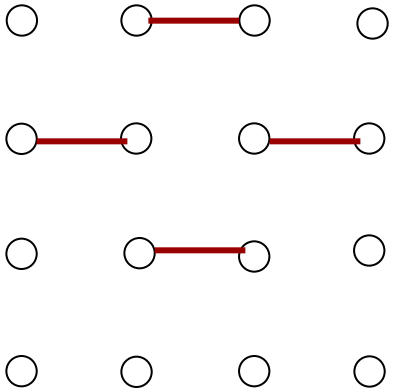


Construindo o casamento máximo....

Algoritmos Aproximados

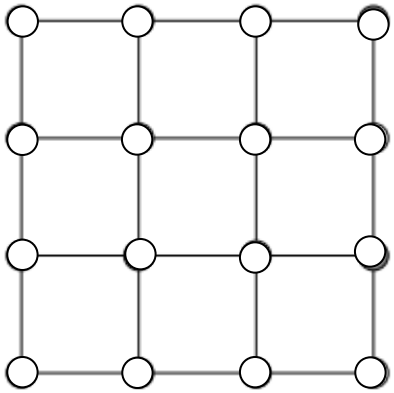


Grafo G

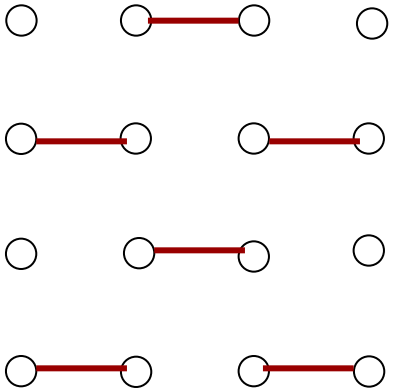


Construindo o casamento máximo....

Algoritmos Aproximados

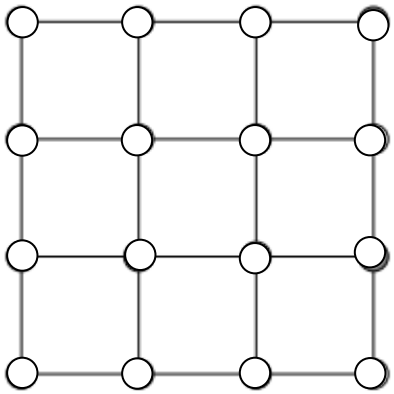


Grafo G

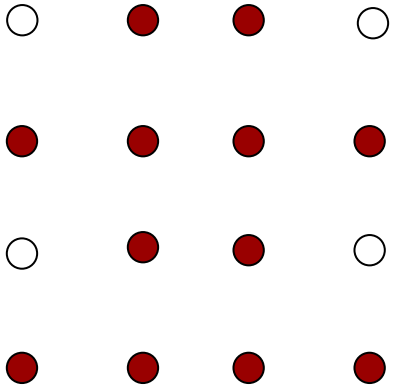


CASAMENTO MÁXIMO!!

Algoritmos Aproximados

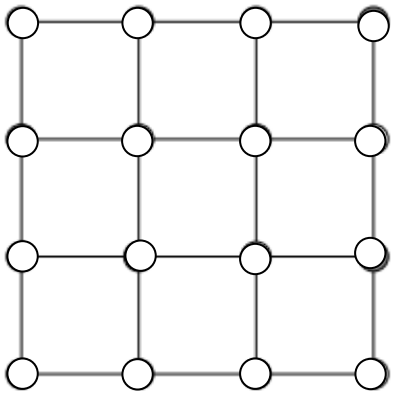


Grafo G

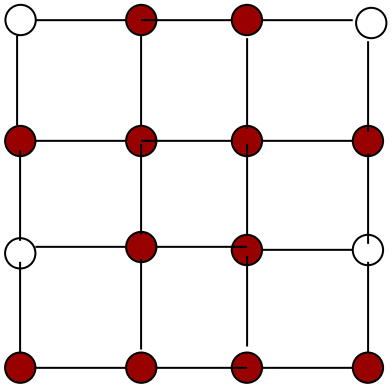


COBERTURA PRODUZIDO PELO ALGORITMO APROXIMADO !!

Algoritmos Aproximados

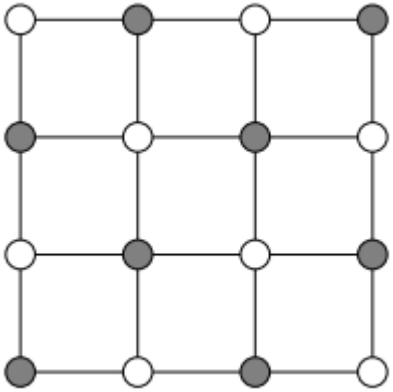


Grafo G



12 vértices !

COBERTURA DE VÉRTICES PRODUZIDO PELO ALGORITMO APROXIMADO !!



SOLUÇÃO ÓTIMA: 8 Vértices