Otimização Combinatória

- Problemas de Otimização Combinatória
- Algoritmos Aproximativos
- Problema da Mochila
- Bin Packing
- Conjunto Independente Máximo

Bibliografia

- B.Korte, J.Vygen. *Combinatorial Optimization: Theory and Algorithms*. 2º edição. Springer, 2002.

- C.H.Papadimitriou, K.Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Pub, 1998

- T.H.Cormen, C.E.Leiserson, R.L.Rivest. *Introduction to Algorithms*. 2º ed. MIT Press, 2001.

- M.R.Garey, D.S.Johnson. *Computers and Intractability: A guide to the Theory of NP-Completeness*. Freeman Co, 1979.

Problemas de Otimização

- Problema Geral de Otimização
  - minimizar $f(x)$, $x \in R^n$
  - Sujeito a $g_i(x) \geq 0$, $i=1...m$

    $h_j(x)=0$, $j=1...p$

- f, g_i, h_j lineares (programação linear)
- Variáveis contínuas (otimização contínua)
- Variáveis discretas (otimização combinatória)

Otimização Combinatória

- Problemas “fáceis”: Caminho, Fluxo e Emparelhamento
- Problemas “difíceis”: Mochila, Bin Packing, Conjunto independente máximo

- Classes P e NP

- Problemas NP-Completos (decisão)
- Problemas NP-Difíceis (otimização)

Algoritmos Aproximativos

- Problema $\wp$, Instância I, Algoritmo A para $\wp$
- $A(I)$: Valor da otimização (min. ou max.)

- Algoritmo A aproximativo por fator k ($k \geq 1$):
  $(1/k)\ OPT(I) \leq A(I) \leq k\ OPT(I)$

- Esquema de aproximação:

  Algoritmo A que recebe como entrada uma instância I de $\wp$ e $\varepsilon>0$ tal que A é algoritmo aproximativo por fator $(1+\varepsilon)$. Complexidade geralmente depende de ε.

Algoritmos Aproximativos

- Alguns problemas podem ser aproximados tanto quanto se queira, por esquemas de aproximação.
- Exemplo: *Problema da Mochila*

- Outros não possuem esquemas de aproximação, mas possuem algoritmos aproximativos por algum fator $k \geq 1$.
- Exemplo: *Cobertura por vértices* ($k = ½$)

- Nem todos os problemas possuem algoritmos aproximativos por algum fator $k \geq 1$.
- Exemplo: *Caixeiro Viajante* (a menos que P=NP)

Knapsack - Problema da Mochila

- maximizar $\sum_{i=1}^{n} v_i x_i$ $\quad v_i \geq 0,\ i=1...n$
- sujeito a $\sum_{j=1}^{n} p_j x_j \leq P$ $\quad p_j \geq 0,\ j=1...n$
- Mochila com capacidade de peso P
- Objetos i com peso p_i e valor v_i

- Fracionário: $x_i \in [0,1]$
- 0-1: $x_i \in \{0,1\}$

Knapsack – Problema Fracionário

- Obtém uma listagem 1,..., n dos objetos pela razão v_i/p_i em ordem decrescente.

- Adiciona os objetos nessa ordem até completar a mochila.

- Exemplo: Ouro (3kg), Prata (2kg), Cobre (5kg)
- Mochila com capacidade para 5,7kg

- Solução: Ouro (3kg), Prata (2kg), Cobre (0,7kg)

Knapsack - Problema 0-1

- Problema NP-Difícil
- Redução polinomial ao Problema da Partição (Instância onde $v_i=p_i$ e P é a metade do peso total) (Decidir se é possível obter valor total > $V=P$)

- Algoritmo PseudoPolinomial
- Programação dinâmica: Complexidade $O(nV)$, onde V é um limite superior para o valor total (p.e., soma dos valores)

Knapsack - Problema 0-1

- Constrói matriz $P(j,k)$, $j=1,...,n$, e $k=1,...,V$.
- $P(j,k)$ é o menor peso total com objetos de 1 a j e valor total igual a k.

- $P(0,0)=0$ e $P(0,k) = \infty$, $k>0$
- $P(j,k) = P(j\text{-}1,k\text{-}v_j) + p_j$, se $v_j \leq k$ e
  $P(j\text{-}1,k\text{-}v_j) + p_j \leq \min\{P, P(j\text{-}1,k)\}$
- $P(j,k) = P(j\text{-}1,k)$, caso contrário

- **Resultado:** $k = \max\{i \in \{0,...,V\}: P(n,i) < \infty\}$

Knapsack - Problema 0-1

- Esquema de aproximação polinomial: Troca precisão por tempo de execução no algoritmo pseudo-polinomial.

- **Idéia:** Divide valores por um fator t e arredonda para baixo. Complexidade $O(nV / t)$, onde $t = \varepsilon V / n$, e V é um limite superior para o valor total. Complexidade final: $O(n^2/\varepsilon)$

- **Detalhes:** Exercício

Bin Packing

- Dada uma lista de n objetos com tamanhos s_i, i=1...n, obter a alocação de todos os objetos utilizando o menor número possível k de caixas com capacidade fixa C.

- Problema NP-Difícil
- Redução polinomial ao Problema da Partição (Instância onde C é metade da soma dos tamanhos) (Decidir se é possível alocar em duas caixas)

Bin Packing

- **Heurísticas gulosas**
- **Next-Fit** (**NF**): Tenta colocar o objeto na caixa corrente. Se não é possível, cria nova caixa
- **First-Fit** (**FF**): Tenta colocar o objeto em "qualquer" uma das caixas já existentes. Se não é possível, cria nova caixa
- **Best-Fit** (**BF**): Tenta colocar o objeto na caixa com menos espaço livre que o caiba. Se não existe, cria nova caixa
- Exemplos: 1, 1, 4, 4. Capacidade 5.

  2, 4, 1, 3. Capacidade 5.

Bin Packing

- **Melhoria:** Escolher objetos com maior tamanho primeiro
- **NFD, FFD e BFD**

- **FF** não deixa duas caixas com menos da metade da capacidade (a segunda caberia na primeira)
- **FF**(I) < 2 **OPT**(I), para quaisquer instâncias I do problema (pior caso: todas as caixas com um *epsilon* acima da metade)

- **FF**(I) ≤ 1.7 **OPT**(I) [Garey, Johnson, 1975]
- **FFD**(I) ≤ 1.222 **OPT**(I) + 1 [Yue, 1990]

Bin Packing

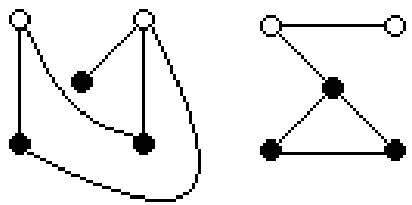
- Instâncias I tais que **FFD**(I) = 1.222 **OPT**(I)
- $I = \{s_1,...,s_{30m}\}$ onde s_i=2+ε (1≤ i ≤6m), s_i=1+2ε (6m≤i≤12m), s_i=1+ε (12m≤i≤18m) e s_i=1-2ε (18m≤i≤30m). Capacidade C=4

- **Solução ótima:** 6m caixas com 2+ε, 1+ε, 1-2ε
  3m caixas com 1+2ε, 1+2ε, 1-2ε, 1-2ε
- **Solução do FFD:** 6m caixas com 2+ε, 1+2ε
  2m caixas com 1+ε, 1+ε, 1+ε
  3m caixas com 1-2ε, 1-2ε, 1-2ε, 1-2ε
- **OPT**(I)=9m, **FFD**(I)=11m

Conjunto Independente Máximo

- Dado um grafo $G = (V,E)$, obter o tamanho do maior conjunto $S \subseteq V$, tal que não existe aresta entre os vértices de S.
- Problema NP-Difícil

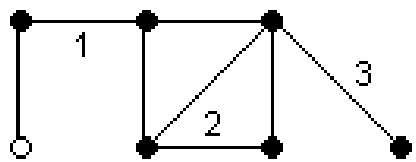
- Se S é um conjunto independente de G, então S é uma clique do complemento de G. Além disso, V-S é uma cobertura por vértices de G.



- Existe algoritmo aproximativo por um fator ½ para o problema da cobertura por vértices.

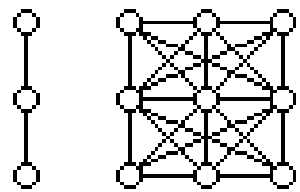
Cobertura por Vértices

- Dado um grafo $G = (V,E)$, obter o tamanho do menor conjunto $C \subseteq V$, tal que toda aresta de G incide em algum vértice de C.
- **Algoritmo aproximativo:** Para cada aresta de G, ponha seus vértices na cobertura C, e remova-os de G, até que G seja vazio.
- Pelo menos um dos vértice de cada passo precisa estar na cobertura ótima. $\text{OPT}(G) \geq \frac{1}{2}|C|$.
- Apesar da forte relação, não é possível aproveitar o algoritmo aproximativo para *Cobertura por vértices*.



Conjunto Independente Máximo

- Seja G^2 o grafo obtido de $G = (V,E)$, com conjunto de vértices $V \times V$, e arestas $((u,u'),(v,v'))$ tais que (u,v) ou $(u',v') \in E$.
- **Lema:** G tem um conjunto independente de tamanho k se e somente se G^2 tem um conjunto independente de tamanho k^2.
- **I** independente em G. $\mathbf{I}^2=\{(u,v): u, v \in \mathbf{I}\}$
- $\mathbf{I}^2$ independente em G^2.
- $\mathbf{I}=\{u: (u,v) \in \mathbf{I}^2$ para algum $v \in \mathbf{I}\}$ ou
- $\mathbf{I}=\{v: (u,v) \in \mathbf{I}^2$ para algum $u \in \mathbf{I}\}$



Conjunto Independente Máximo

- **Teorema:** Se existe um algoritmo aproximativo de fator k, então existe um esquema de aproximação polinomial.

- G G^2
- $\geq k^{-1}\,\mathrm{OPT}^2(G)$
- $\geq k^{-1/2}\,\mathrm{OPT}(G)$ $\geq k^{-1/2}\,\mathrm{OPT}(G)$
- $\geq k^{-1/4}\,\mathrm{OPT}(G)$ $\geq k^{-1/4}\,\mathrm{OPT}(G)$