1 Aproximativos - AULA 1 - Introdução

2 Aproximativos - AULA 2 - Bin Packing

BIN PACKING: **Instância:** n itens de pesos $0 < p_1, p_2, \ldots, p_n \leq 1$. **Objetivo:** Obter o menor número de sacolas (bins) de capacidade 1 para empacotar todos os itens.

Algoritmo FIRST-FIT (FF): Para cada item, tente colocá-lo em alguma sacola existente. Caso não seja possível, crie uma sacola para o item.

Lemma 2.1.

$$OPT \geq \left\lceil \sum_{i=1}^{n} p_i \right\rceil$$

Demonstração. O peso total dos itens é $\sum_{i=1}^{n} p_i$. Se fosse permitido dividir os itens, teríamos exatamente $\lceil \sum_{i=1}^{n} p_i \rceil$ sacolas. □

Theorem 2.2. *FF é 2-aproximativo.*

Demonstração. Seja B o número de sacolas usadas pelo FF. Então pelo menos $B - 1$ sacolas estão cheias pela metade. Logo

$$\sum_{i=1}^{n} p_i > \frac{B-1}{2}.$$

Logo $B - 1 < 2 \cdot OPT$ e consequentemente $B \leq 2 \cdot OPT$. □

[Garey, Johnson, 1974] - [Dósa, Sgall, 2013]: **FF é 1.7-aproximativo**

Instância ótima:

- 20 itens de peso $1/42 - 3\varepsilon$
- 20 itens de peso $1/7 + \varepsilon$
- 20 itens de peso $1/3 + \varepsilon$
- 20 itens de peso $1/2 + \varepsilon$

OPT: 20 sacolas com itens $1/2 + \varepsilon$, $1/3 + \varepsilon$, $1/7 + \varepsilon$ e $1/42 - 3\varepsilon$.

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{7} + \frac{1}{42} = \frac{21}{42} + \frac{14}{42} + \frac{6}{42} + \frac{1}{42} = \frac{42}{42} = 1$$

FF: 34 sacolas

- 1 sacola com 20 itens $1/42 - 3\varepsilon$ e 3 itens $1/7 + \varepsilon$: $\quad \frac{20}{42} + \frac{3}{7} = \frac{10}{21} + \frac{9}{21} = \frac{19}{21}$
- 2 sacolas com 6 itens $1/7 + \varepsilon$
- 1 sacola com 5 itens $1/7 + \varepsilon$
- 10 sacolas com 2 itens $1/3 + \varepsilon$
- 20 sacolas com 1 item $1/2 + \varepsilon$

Algoritmo FIRST-FIT-DECREASING (FF): Ordene os itens do maior para o menor peso. Aplique o Algoritmo FF.

Theorem 2.3. *FFD é 1.5-aproximativo.*

Demonstração. Considere os itens já ordenados do maior para o menor peso. Considere o empacotamento do FFD em k sacolas numeradas de 1 a k. Queremos mostrar que $k \leq 1.5 \cdot OPT$. Seja $P = \sum_{i=1}^{n} p_i$ o peso total dos itens. Pelo lema anterior, $OPT \geq P$.

Seja $1 \leq b \leq k$ uma sacola qualquer. Se a sacola b tem um item i com $p_i > 1/2$, então toda sacola anterior tem exatamente 1 item. Logo, $OPT \geq b$.

Suponha que a sacola b NÃO tem um item com peso maior que 1/2. Então toda sacola de b a $k-1$ tem pelo menos 2 itens, totalizando pelo menos $2(k-b)$ itens (que não cabem nas sacolas anteriores).

Caso 1: se $b \leq 2(k-b)$, então, adicionando à cada sacola anterior a b um desses $2(k-b)$ itens, obtemos $b-1$ sacolas acima do peso. Logo $P > b-1$.

Caso 2: Se $b > 2(k-b)$, então, adicionando cada um dos $2(k-b)$ itens a uma sacola anterior diferente, obtemos $2(k-b)$ sacolas acima do peso. Logo, $P > 2(k-b)$.

Portanto, $OPT \geq b$ ou $OPT > 2(k-b)$ para toda sacola $1 \leq b \leq k$. Maximizando o mínimo: $b = 2(k-b) \implies b = 2k/3$. Tomando $b = \lceil 2k/3 \rceil$, temos $OPT \geq 2k/3$ ou $OPT > 2(k - \lceil 2k/3 \rceil) \geq 2k/3$. PORTANTO, $k = (3/2) \cdot OPT$. □

Theorem 2.4. *Para qualquer $\varepsilon > 0$, não existe algoritmo polinomial $(1.5-\varepsilon)$-aproximativo, a menos que P=NP.*

Demonstração. Problema clássico da PARTIÇÃO: dados $s_1, s_2, s_n \geq 0$, é possível dividí-los em dois subconjuntos com soma igual?

PARTIÇÃO é NP-Completo (SUBSET-SUM é uma generalização de PARTIÇÃO).

Seja $S = \sum_{i=1}^{n} s_i$ e, para todo $i = 1, \ldots, n$, seja $p_i = 2 \cdot s_i / S$. Logo, $P = \sum_{i=1}^{n} p_i = 2$.

PARTIÇÃO é SIM $\iff$ pesos $p_1, \ldots, p_n$ cabem em duas sacolas $\iff$ um algoritmo $(1.5-\varepsilon)$-aproximativo obtém a solução exata: $(1.5-\varepsilon) \cdot 2 = 3 - 2\varepsilon < 3$. Impossível, se P≠NP. □

[Yue, 1991]: **FFD**$\leq (11/9) \cdot OPT + 1$. Fator assintótico $1.222\ldots$

Instância ótima:

- $6m$ itens de peso $1/2 + \varepsilon$
- $6m$ itens de peso $1/4 + 2\varepsilon$
- $6m$ itens de peso $1/4 + \varepsilon$
- $12m$ itens de peso $1/4 - 2\varepsilon$

OPT: 9 sacolas

- $6m$ sacolas com itens $1/2 + \varepsilon$, $1/4 + \varepsilon$ e $1/4 - 2\varepsilon$;
- $3m$ sacolas com 2 itens $1/4 + 2\varepsilon$ e 2 itens $1/4 - 2\varepsilon$.

FF: 11 sacolas

- $6m$ sacolas com itens $1/2 + \varepsilon$ e $1/4 + 2\varepsilon$;
- $2m$ sacolas com 3 itens $1/4 + \varepsilon$;
- $3m$ sacolas com 4 itens $1/4 - 2\varepsilon$.

3 Aproximativos - AULA 3a - Escalonamento

ESCALONAMENTO: **Instância:** m máquinas idênticas e n tarefas com tempos de execução $t_1, \ldots, t_n > 0$. **Objetivo:** Atribuir tarefas às máquinas minimiizando o tempo máximo de operação de qualquer uma das máquinas.

Problema NP-Difícil até mesmo para $m = 2$ máquinas (Redução do problema da PARTIÇÃO).

Escalonamento-de-Graham: Para cada tarefa, atribua-a à máquina cujo tempo total é mínimo.

Exemplo: tempos $t = [4, 2, 1, 5, 9, 2, 6]$ em $m = 3$ máquinas

- **Graham:** $M_1 = 4 + 2 + 6 = \mathbf{12}$; $M_2 = 2 + 9 = 11$ $M_3 = 1 + 5 = 6$.
- **OPT:** $M_1 = 4 + 6 = \mathbf{10}$; $M_2 = 2 + 5 + 2 = 9$ $M_3 = 1 + 9 = \mathbf{10}$.
  É ótimo pois $\frac{1}{m} \sum_{i=1}^{n} t_i = (4 + 2 + 1 + 5 + 9 + 2 + 6)/3 = 29/3 = 9.666\ldots$

Lemma 3.1. *$OPT \geq \max_i t_i$ e $OPT \geq \frac{1}{m} \sum_{i=1}^{n} t_i$*

Demonstração. A maior tarefa terá que ser executada em alguma máquina. Além disso, se pudéssemos dividir igualmente as tarefas entre as máquinas, teríamos a média em cada. □

Theorem 3.2. *O Escalonamento-de-Graham é 2-aproximativo. Se $m = O(n)$, então o algoritmo é polinomial.*

Demonstração. Seja T_F o tempo final do escalonamento, seja j a máquina que atingiu esse tempo e seja i a última tarefa da máquina j. Seja T o tempo da máquina j antes da tarefa i: $T_F = T + t_i$.

Pelo algoritmo, T é o menor tempo entre as máquinas na atribuição da tarefa i. Portanto $T \cdot m \leq \sum_{i=1}^{n} t_i$. Pelo lema, $T \leq OPT$, pelo lema.

Assim, $T_F = T + t_i \leq 2 \cdot OPT$. □

4 Aproximativos - AULA 3b - Caixeiro Viajante

CAIXEIRO VIAJANTE: **Instância:** Grafo completo não-direcionado com pesos nas arestas. **Objetivo:** Obter menor ciclo passando por todos os vértices exatamente uma vez.

Problema NP-Difícil.

Theorem 4.1. *Não existe algoritmo α-aproximativo para Caixeiro-Viajante, para nenhum valor de $\alpha > 0$, a menos que P=NP.*

Demonstração. □

5 Aproximativos - AULA 4 - Caixeiro Viajante Métrico

CAIXEIRO VIAJANTE MÉTRICO: **Instância:** Grafo completo não-direcionado com pesos nas arestas, respeitando a desigualdade triangular. **Objetivo:** Obter menor ciclo passando por todos os vértices exatamente uma vez.

Problema NP-Difícil.

Algoritmo RSL (Rosenkratz-Stearns-Lewis), 1977:

Lemma 5.1. *Algoritmo RSL é 2-aproximativo.*

Demonstração. □

Algoritmo Christofides, 1976:

Lemma 5.2. *Algoritmo Christofides é 1.5-aproximativo.*

Demonstração. □

6 Aproximativos - AULA 5 - Mochila

MOCHILA: **Instância:** n itens $1, \ldots, n$ com pesos $p_1, \ldots, p_n$ e valores $v_1, \ldots, v_n$. Mochila com capacidade C. **Objetivo:** Obter subconjunto S de itens tal que $\sum_{i \in S} p_i \leq C$ e $\sum_{i \in S} v_i$ seja máxima.

Problema NP-Difícil (Redução do problema SUBSET-SUM).

Programação dinâmica Mochila1: Seja $V[i,k]$ o valor máximo com mochila de capacidade $k \leq C$, considerando apenas itens de 1 a i. **Objetivo:** Calcular $V[n,C]$. **Tempo:** $\Theta(n \cdot C)$ pseudo-polinomial.

$$V[i,k] \;=\; \begin{cases} 0, & \text{se } i = 0, \\ V[i-1,k], & \text{se } i > 0 \text{ e } p_i > k, \\ \max\{V[i-1,k], V[i-1,k-p_i] + v_i\}, & \text{se } i > 0 \text{ e } p_i \leq k. \end{cases}$$

Programação dinâmica Mochila2: Seja $P[i,\ell]$ o peso mínimo para atingir valor exatamente ℓ, considerando apenas itens de 1 a i. $P[i,\ell] = \infty$ se impossível. **Objetivo:** Calcular $\max\{\ell \,:\, P[n,\ell] \leq C\}$. **Limite superior:** $0 \leq \ell \leq n \cdot v^*$, onde v^* é o valor do item mais valioso. **Tempo:** $\Theta(n^2 \cdot v^*)$ pseudo-polinomial.

$$P[i,\ell] \;=\; \begin{cases} 0, & \text{se } i = 0 \text{ e } \ell = 0, \\ \infty, & \text{se } i = 0 \text{ e } \ell > 0, \\ P[i-1,\ell], & \text{se } i > 0 \text{ e } v_i > \ell, \\ \min\{P[i-1,\ell], P[i-1,\ell-v_i] + p_i\}, & \text{se } i > 0 \text{ e } v_i \leq \ell. \end{cases}$$

Algoritmo aproximativo Mochila3(ε): Dado $\varepsilon > 0$, seja $R = \varepsilon v^*/n$. Para cada item i, seja $v'_i = \lfloor v_i/R \rfloor$. Retorne o valor da Programação Dinâmica 2 para pesos p, valores v' e capacidade C.

Lemma 6.1. *Mochila3(ε) é $(1-\varepsilon)$-aproximativo e tem tempo $\Theta(\frac{1}{\varepsilon}n^3)$.*

Demonstração. Seja O o conjunto ótimo: $opt = \sum_{i \in O} v_i$. Temos que:

$$v'_i > \frac{v_i}{R} - 1 \implies R \cdot \sum_{i \in O} v'_i > \sum_{i \in O} v_i - R \cdot |O| \implies R \cdot \sum_{i \in O} v'_i > \sum_{i \in O} v_i - R \cdot n$$

Seja S' o conjunto retornado por Mochila2.

$$\sum_{i \in S'} v_i \geq R \cdot \sum_{i \in S'} v'_i \geq R \cdot \sum_{i \in O} v'_i \geq \sum_{i \in O} v_i - nR \;=\; opt - \varepsilon v^* \geq (1-\varepsilon) opt,$$

pois $v^* \leq opt$. □

FPTAS: Full Polynomial Time Approximation Scheme (Esquema de Aproximação de Tempo Completamente Polinomial): Polinomial em n e em $1/\varepsilon$.

7 Aproximativos - AULA 6 - Set Cover

SET COVER: **Instância:** Conjunto $U = \{u_1, \ldots, u_n\}$, k subconjuntos $S_1, \ldots, S_k \subseteq U$ cada um com um custo $c_1, \ldots, c_k$, respectivamente. **Objetivo:** Encontrar $B \subseteq \{1, \ldots, k\}$ tal que $\bigcup_{j \in B} S_j = U$ e $\sum_{j \in B} c_j$ seja mínima.

Problema NP-Difícil (Redução do problema VERTEX-COVER).

Algoritmo Guloso: Seja $C = \emptyset$.

Theorem 7.1. *Algoritmo Guloso do Set Cover tem fator de aproximação $H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$. Lembre que $H_n \leq 1 + \ln n$.*

Demonstração. □