Algoritmos de Aproximação

Segundo Semestre de 2012

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila: conjunto $S \subseteq \{1, \ldots, n\} = [n]$ tal que $\sum_{i \in S} s_i \leq B$.

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila: conjunto $S \subseteq \{1, \ldots, n\} = [n]$ tal que $\sum_{i \in S} s_i \leq B$.

O valor de uma mochila é o número $\sum_{i \in S} v_i$.

Uma mochila é ótima se tem valor máximo.

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila: conjunto $S \subseteq \{1, \ldots, n\} = [n]$ tal que $\sum_{i \in S} s_i \leq B$.

O valor de uma mochila é o número $\sum_{i \in S} v_i$.

Uma mochila é ótima se tem valor máximo.

Problema (Knapsack Problem): Dados (n, s, v, B), encontrar uma mochila ótima.

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila: conjunto $S \subseteq \{1, \ldots, n\} = [n]$ tal que $\sum_{i \in S} s_i \leq B$.

O valor de uma mochila é o número $\sum_{i \in S} v_i$.

Uma mochila é ótima se tem valor máximo.

Problema (Knapsack Problem): Dados (n, s, v, B), encontrar uma mochila ótima.

Exemplo: $B = 50$, $n = 4$

	1	2	3	4
s	40	30	20	10
v	840	600	400	100

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila: conjunto $S \subseteq \{1, \ldots, n\} = [n]$ tal que $\sum_{i \in S} s_i \leq B$.

O valor de uma mochila é o número $\sum_{i \in S} v_i$.

Uma mochila é ótima se tem valor máximo.

Problema (Knapsack Problem): Dados (n, s, v, B), encontrar uma mochila ótima.

Exemplo: $B = 50$, $n = 4$

	1	2	3	4
s	40	30	20	10
v	840	600	400	100

Mochila

Suponha dado um número inteiro não-negativo B e vetores de inteiros positivos $s[1\,..\,n]$ e $v[1\,..\,n]$.

Mochila: conjunto $S \subseteq \{1, \ldots, n\} = [n]$ tal que $\sum_{i \in S} s_i \leq B$.

O valor de uma mochila é o número $\sum_{i \in S} v_i$.

Uma mochila é ótima se tem valor máximo.

Problema (Knapsack Problem): Dados (n, s, v, B), encontrar uma mochila ótima.

Exemplo: $B = 50$, $n = 4$

	1	2	3	4
s	40	30	20	10
v	840	600	400	100

Algoritmo de programação dinâmica

Devolve o valor de uma mochila ótima para (n, s, v, B).
Suponha que $v_i \leq B$ para todo i.

MOCHILA-PD (n, s, v, B)

1 $A[1] \leftarrow \{(0,0), (s_1, v_1)\}$
2 **para** $j \leftarrow 2$ **até** n **faça**
3 $\quad A[j] \leftarrow A[j-1]$
4 $\quad$ **para cada** (t, w) em $A[j-1]$ **faça**
5 $\quad\quad$ **se** $t + s_j \leq B$
6 $\quad\quad\quad$ **então** $A[j] \leftarrow A[j] \cup \{(t + s_j, w + v_j)\}$
7 $\quad$ remova os pares dominados de $A[j]$
8 **devolva** w para (t, w) em $A[n]$ com w máximo

Algoritmo de programação dinâmica

Devolve o valor de uma mochila ótima para (n, s, v, B).
Suponha que $v_i \leq B$ para todo i.

```
MOCHILA-PD (n, s, v, B)
 1   A[1] ← {(0, 0), (s_1, v_1)}
 2   para j ← 2 até n faça
 3         A[j] ← A[j − 1]
 4         para cada (t, w) em A[j − 1] faça
 5               se t + s_j ≤ B
 6                     então A[j] ← A[j] ∪ {(t + s_j, w + v_j)}
 7         remova os pares dominados de A[j]
 8   devolva w para (t, w) em A[n] com w máximo
```

Consumo de tempo é $\mathrm{O}(n \min\{B, V\})$, onde $V = \sum_i v_i$.

Algoritmo de programação dinâmica

Devolve o valor de uma mochila ótima para (n, s, v, B).
Suponha que $v_i \leq B$ para todo i.

```
MOCHILA-PD (n, s, v, B)
1  A[1] ← {(0,0), (s_1, v_1)}
2  para j ← 2 até n faça
3       A[j] ← A[j − 1]
4       para cada (t, w) em A[j − 1] faça
5            se t + s_j ≤ B
6                 então A[j] ← A[j] ∪ {(t + s_j, w + v_j)}
7       remova os pares dominados de A[j]
8  devolva w para (t, w) em A[n] com w máximo
```

Consumo de tempo é $\mathrm{O}(n \ \min\{B, V\})$, onde $V = \sum_i v_i$.

Adapte o algoritmo para devolver uma mochila ótima.

FPTAS usando arredondamento

Fixe $\epsilon > 0$.

Devolve uma mochila para (n, s, v, B) de valor $\geq (1 - \epsilon)\mathrm{OPT}$.

Suponha que $v_i \leq B$ para todo i.

MOCHILA-ARRED$_\epsilon$ (n, s, v, B)

1 $M \leftarrow \max_i v_i$
2 $\mu \leftarrow \epsilon M/n$
3 **para** $i \leftarrow 1$ **até** n **faça**
4 $\quad v'_i \leftarrow \lfloor v_i/\mu \rfloor$
5 **devolva** MOCHILA-PD (n, s, v', B)

FPTAS usando arredondamento

Fixe $\epsilon > 0$.

Devolve uma mochila para (n, s, v, B) de valor $\geq (1 - \epsilon)\mathrm{OPT}$.

Suponha que $v_i \leq B$ para todo i.

MOCHILA-ARRED$_\epsilon$ (n, s, v, B)

1 $M \leftarrow \max_i v_i$
2 $\mu \leftarrow \epsilon M/n$
3 **para** $i \leftarrow 1$ **até** n **faça**
4 $v'_i \leftarrow \lfloor v_i/\mu \rfloor$
5 **devolva** MOCHILA-PD (n, s, v', B)

Consumo de tempo é $\mathrm{O}(n^3/\epsilon)$.

FPTAS usando arredondamento

Fixe $\epsilon > 0$.

Devolve uma mochila para (n, s, v, B) de valor $\geq (1 - \epsilon)\mathrm{OPT}$.

Suponha que $v_i \leq B$ para todo i.

MOCHILA-ARRED$_\epsilon$ (n, s, v, B)

1 $M \leftarrow \max_i v_i$
2 $\mu \leftarrow \epsilon M/n$
3 **para** $i \leftarrow 1$ **até** n **faça**
4 $v'_i \leftarrow \lfloor v_i/\mu \rfloor$
5 **devolva** MOCHILA-PD (n, s, v', B)

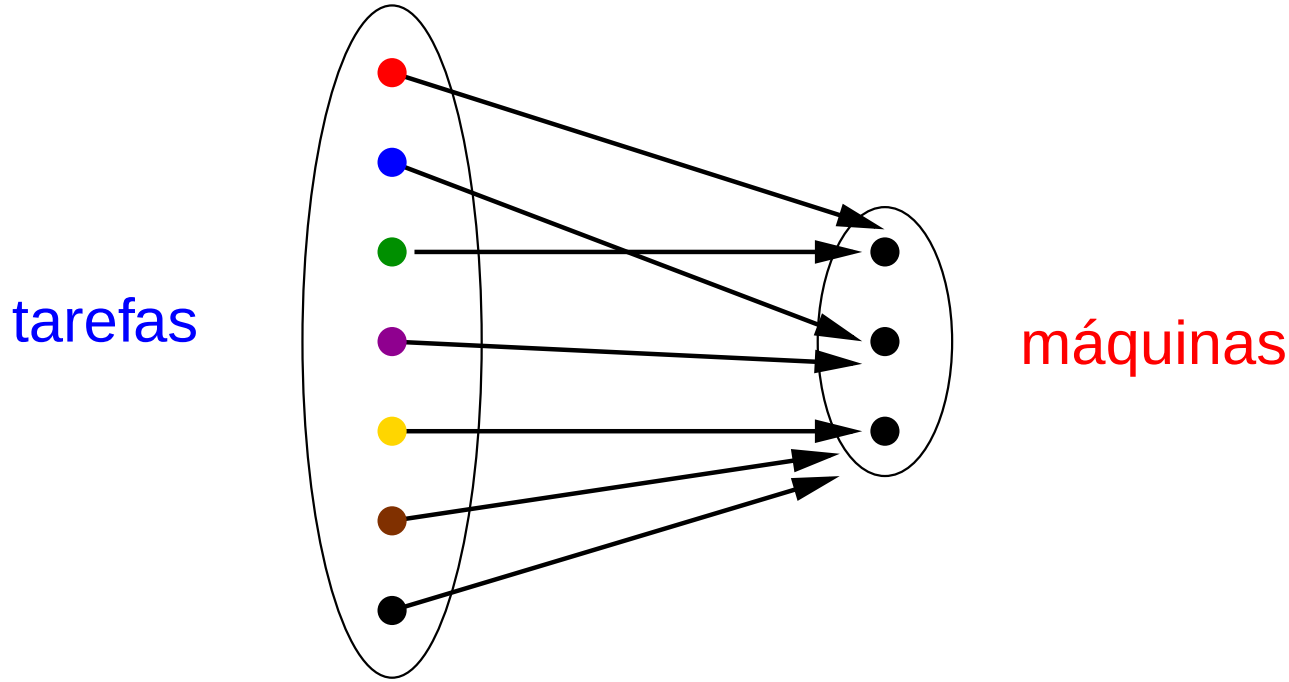
Consumo de tempo é $\mathrm{O}(n^3/\epsilon)$.

Teorema: {MOCHILA-ARRED$_\epsilon$} é um FPTAS para o problema da mochila.

Escalonamento de máquinas idênticas

Dados: m máquinas
n tarefas ($[n] = \{1, \ldots, n\}$)
duração $d[i]$ da tarefa i ($i = 1, \ldots, n$)

um escalonamento é uma partição $\{M[1], \ldots, M[m]\}$ de $[n]$



Encontrar escalonamento com tempo de conclusão mínimo.

Ideia para um algoritmo

Divida as tarefas em longas e curtas.

Ideia para um algoritmo

Divida as tarefas em longas e curtas.

Tarefa curta: $d_\ell \leq \frac{1}{km} \sum_{j=1}^{n} d_j$

Fato: Há no máximo km tarefas longas.

Ideia para um algoritmo

Divida as tarefas em longas e curtas.

Tarefa curta: $d_\ell \leq \frac{1}{km} \sum_{j=1}^{n} d_j$

Fato: Há no máximo km tarefas longas.

Encontre por enumeração um escalonamento ótimo das tarefas longas.

Ideia para um algoritmo

Divida as tarefas em longas e curtas.

Tarefa curta: $d_\ell \leq \frac{1}{km} \sum_{j=1}^{n} d_j$

Fato: Há no máximo km tarefas longas.

Encontre por enumeração um escalonamento ótimo das tarefas longas.

Aplique Graham às tarefas curtas, a partir do escalonamento das tarefas longas.

Ideia para um algoritmo

Divida as tarefas em longas e curtas.

Tarefa curta: $d_\ell \leq \frac{1}{km} \sum_{j=1}^{n} d_j$

Fato: Há no máximo km tarefas longas.

Encontre por enumeração um escalonamento ótimo das tarefas longas.

Aplique Graham às tarefas curtas, a partir do escalonamento das tarefas longas.

Quanto tempo consome este algoritmo?

Qual é a razão de aproximação?

Consumo de tempo

Para especificar cada escalonamento das tarefas longas, basta dizer a qual máquina cada tarefa foi escalonada.
(A ordem nas máquinas não importa.)

Consumo de tempo

Para especificar cada escalonamento das tarefas longas, basta dizer a qual máquina cada tarefa foi escalonada.
(A ordem nas máquinas não importa.)

Logo há no máximo m^{km} escalonamentos
para as tarefas longas.

Consumo de tempo

Para especificar cada escalonamento das tarefas longas, basta dizer a qual máquina cada tarefa foi escalonada.
(A ordem nas máquinas não importa.)

Logo há no máximo m^{km} escalonamentos
para as tarefas longas.

Se m é constante, isso é polinomial
e podemos usar força-bruta.

Consumo de tempo

Para especificar cada escalonamento das tarefas longas, basta dizer a qual máquina cada tarefa foi escalonada.
(A ordem nas máquinas não importa.)

Logo há no máximo m^{km} escalonamentos
para as tarefas longas.

Se m é constante, isso é polinomial
e podemos usar força-bruta.

O resultado é polinomial neste caso.

Qualidade do escalonamento produzido

Seja ℓ a tarefa que termina por último no escalonamento.

Qualidade do escalonamento produzido

Seja ℓ a tarefa que termina por último no escalonamento.

Se a tarefa ℓ é longa,
então claro que o escalonamento produzido é ótimo.

Qualidade do escalonamento produzido

Seja ℓ a tarefa que termina por último no escalonamento.

Se a tarefa ℓ é longa,
então claro que o escalonamento produzido é ótimo.

Se a tarefa ℓ é curta,
a mesma análise do algoritmo de Graham se aplica!

Ou seja, $C_{\max} \leq d_\ell + \frac{1}{m} \sum_{j \neq \ell} d_j$.

Qualidade do escalonamento produzido

Seja ℓ a tarefa que termina por último no escalonamento.

Se a tarefa ℓ é longa,
então claro que o escalonamento produzido é ótimo.

Se a tarefa ℓ é curta,
a mesma análise do algoritmo de Graham se aplica!

Ou seja, $C_{\max} \leq d_\ell + \frac{1}{m} \sum_{j \neq \ell} d_j$.

Como $d_\ell \leq \frac{1}{km} \sum_{j=1}^{n} d_j$, temos que

$$C_{\max} \leq \frac{1}{km} \sum_{j=1}^{n} d_j + \frac{1}{m} \sum_{j \neq \ell} d_j \leq \left(1+\frac{1}{k}\right) \frac{\sum_{j=1}^{n} d_j}{m} \leq \left(1+\frac{1}{k}\right) \mathrm{OPT}.$$

Conclusão

Suponha que m é uma constante.

ESCALONAMENTO$_k$ (n, d)

1 $t \leftarrow 0 \quad q \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n **faça**
3 **se** $d_i > \frac{1}{km} \sum_{j=1}^{n} d_j$
4 **então** $t \leftarrow t + 1 \quad d'_t \leftarrow d_i$
5 **senão** $q \leftarrow q + 1 \quad d''_q \leftarrow d_i$
6 $M \leftarrow$FORÇA-BRUTA$_m(t, d')$
7 **devolva** GRAHAM (M, m, q, d'')

Conclusão

Suponha que m é uma constante.

ESCALONAMENTO$_k$ (n, d)

1 $t \leftarrow 0 \quad q \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n **faça**
3 **se** $d_i > \frac{1}{km} \sum_{j=1}^{n} d_j$
4 **então** $t \leftarrow t + 1 \quad d'_t \leftarrow d_i$
5 **senão** $q \leftarrow q + 1 \quad d''_q \leftarrow d_i$
6 $M \leftarrow$FORÇA-BRUTA$_m(t, d')$
7 **devolva** GRAHAM (M, m, q, d'')

GRAHAM (M, m, q, d''): aplica Graham a partir de M.

Conclusão

Suponha que m é uma constante.

ESCALONAMENTO$_k$ (n, d)

1 $t \leftarrow 0 \quad q \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** n **faça**
3 **se** $d_i > \frac{1}{km} \sum_{j=1}^{n} d_j$
4 **então** $t \leftarrow t + 1 \quad d'_t \leftarrow d_i$
5 **senão** $q \leftarrow q + 1 \quad d''_q \leftarrow d_i$
6 $M \leftarrow$FORÇA-BRUTA$_m(t, d')$
7 **devolva** GRAHAM (M, m, q, d'')

GRAHAM (M, m, q, d''): aplica Graham a partir de M.

Teorema: ESCALONAMENTO$_k$ é um PTAS para o problema do escalonamento em máquinas idênticas quando o número de máquinas é constante.