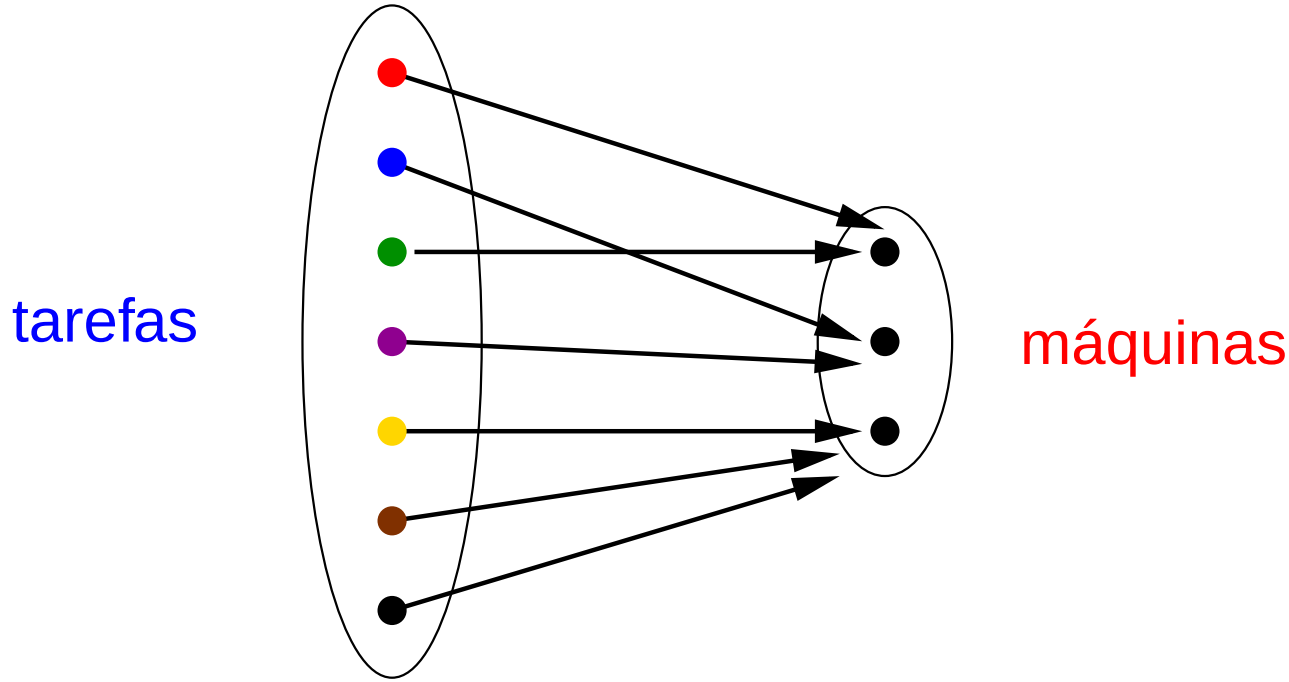
# Algoritmos de Aproximação

## Segundo Semestre de 2012

# 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 $\ell$ a tarefa que termina por último no escalonamento.

# Qualidade do escalonamento produzido

Seja $\ell$ a tarefa que termina por último no escalonamento.

Se a tarefa $\ell$ é longa,
então claro que o escalonamento produzido é ótimo.

# Qualidade do escalonamento produzido

Seja $\ell$ a tarefa que termina por último no escalonamento.

Se a tarefa $\ell$ é longa,
então claro que o escalonamento produzido é ótimo.

Se a tarefa $\ell$ 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 $\ell$ a tarefa que termina por último no escalonamento.

Se a tarefa $\ell$ é longa,
então claro que o escalonamento produzido é ótimo.

Se a tarefa $\ell$ 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 da aula passada

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 da aula passada

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 da aula passada

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.

# Algoritmo auxiliar

$T$: um valor estimado de makespan $\quad (T \geq \sum_{i=1}^{n} \frac{d_i}{m})$

Algoritmo $B_k$:
ou não há escalonamento com makespan $\leq T$
ou devolve escalonamento com makespan $\leq (1 + \frac{1}{k})T$.

# Algoritmo auxiliar

$T$: um valor estimado de makespan $\quad (T \geq \sum_{i=1}^{n} \frac{d_i}{m})$

$t$-escalonamento: escalonamento com makespan $\leq t$.

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

# Algoritmo auxiliar

$T$: um valor estimado de makespan $\quad (T \geq \sum_{i=1}^{n} \frac{d_i}{m})$

$t$-escalonamento: escalonamento com makespan $\leq t$.

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

tarefa longa: $d_\ell > T/k$.

para cada tarefa longa $i$ faça
seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

# Algoritmo auxiliar

$T$: um valor estimado de makespan $\quad (T \geq \sum_{i=1}^{n} \frac{d_i}{m})$

$t$-escalonamento: escalonamento com makespan $\leq t$.

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

tarefa longa: $d_\ell > T/k$.

para cada tarefa longa $i$ faça
seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
então devolva **NÃO**
senão considere o escalonamento das tarefas longas
com $d_i$ e estenda com Graham nas tarefas curtas.

# Algoritmo auxiliar

$T$: um valor estimado de makespan $\quad (T \geq \sum_{i=1}^{n} \frac{d_i}{m})$

$t$-escalonamento: escalonamento com makespan $\leq t$.

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

tarefa longa: $d_\ell > T/k$.

para cada tarefa longa $i$ faça
seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
então devolva **NÃO**
senão considere o escalonamento das tarefas longas
com $d_i$ e estenda com Graham nas tarefas curtas.

Vamos mostrar que $B_k$ faz o que promete.

# Algoritmo auxiliar

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

# Algoritmo auxiliar

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

Se há $T$-escalonamento com $d_i$
  então há com $d'_i$ para as tarefas longas.

# Algoritmo auxiliar

Algoritmo $B_k$: ou não há $T$-escalonamento
ou devolve $(1 + \frac{1}{k})T$-escalonamento.

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

Se há $T$-escalonamento com $d_i$
  então há com $d'_i$ para as tarefas longas.

Como $d'_i \geq T/k$, há no máximo $k$ tarefas por máquina.
Como $d_i - d'_i < T/k^2$, o makespan do correspondente
escalonamento com $d_i$ é no máximo $T + T/k$.

# Algoritmo auxiliar

Algoritmo $B_k$:

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

# Algoritmo auxiliar

Algoritmo $B_k$:

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

Há um $(1 + \frac{1}{k})T$-escalonamento das tarefas longas com $d_i$.

# Algoritmo auxiliar

Algoritmo $B_k$:

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

Há um $(1 + \frac{1}{k})T$-escalonamento das tarefas longas com $d_i$.

Seja $\ell$ a última tarefa a concluir.

Se $\ell$ é longa, é $(1 + \frac{1}{k})T$-escalonamento.

# Algoritmo auxiliar

Algoritmo $B_k$:

para cada tarefa longa $i$ (com $d_i > T/k$) faça
  seja $d'_i$ o maior múltiplo de $T/k^2 \leq d_i$

se não há $T$-escalonamento das tarefas longas com $d'_i$
  então devolva **NÃO**
  senão considere o escalonamento das tarefas longas
    com $d_i$ e estenda com Graham nas tarefas curtas.

Há um $(1 + \frac{1}{k})T$-escalonamento das tarefas longas com $d_i$.

Seja $\ell$ a última tarefa a concluir.

Se $\ell$ é longa, é $(1 + \frac{1}{k})T$-escalonamento.

Se $\ell$ é curta,

$$d_\ell + \frac{\sum_{j \neq \ell} d_j}{m} \leq \frac{T}{k} + T = (1 + \frac{1}{k})T.$$

# PD para as tarefas longas

Se há $d'_\ell > T$, não há $T$-escalonamento.

# PD para as tarefas longas

Se há $d'_\ell > T$, não há $T$-escalonamento.

Senão instância é dada por vetor $k^2$-dimensional, onde componente $i$ é o número de tarefa $\ell$ com $d'_\ell = i\frac{T}{k^2}$.

# PD para as tarefas longas

Se há $d'_\ell > T$, não há $T$-escalonamento.

Senão instância é dada por vetor $k^2$-dimensional, onde componente $i$ é o número de tarefa $\ell$ com $d'_\ell = i\frac{T}{k^2}$.

No máximo $n^{k^2}$ instâncias diferentes.
(polinomial pois $k$ é fixo!)

# PD para as tarefas longas

Se há $d'_\ell > T$, não há $T$-escalonamento.

Senão instância é dada por vetor $k^2$-dimensional, onde componente $i$ é o número de tarefa $\ell$ com $d'_\ell = i\frac{T}{k^2}$.

No máximo $n^{k^2}$ instâncias diferentes.
(polinomial pois $k$ é fixo!)

Cada máquina tem no máximo $k$ tarefas (longas).
Vetor $k^2$-dimensional, $(k+1)^{k^2}$ tipos possíveis.

# PD para as tarefas longas

Se há $d'_\ell > T$, não há $T$-escalonamento.

Senão instância é dada por vetor $k^2$-dimensional, onde componente $i$ é o número de tarefa $\ell$ com $d'_\ell = i\frac{T}{k^2}$.

No máximo $n^{k^2}$ instâncias diferentes.
(polinomial pois $k$ é fixo!)

Cada máquina tem no máximo $k$ tarefas (longas).
Vetor $k^2$-dimensional, $(k+1)^{k^2}$ tipos possíveis.

Configuração de máquina: $(s_1, \ldots, s_{k^2})$ tq

$$\sum_{i=1}^{k^2} s_i \frac{iT}{k^2} \leq T.$$

$\mathcal{C}$: conjunto das configurações de máquina.

# PD para as tarefas longas

$\mathrm{opt}(n_1, \ldots, n_{k^2})$: número mínimo de máquinas suficiente para escalonar a instância $(n_1, \ldots, n_{k^2})$.

# PD para as tarefas longas

$\mathrm{opt}(n_1, \ldots, n_{k^2})$: número mínimo de máquinas suficiente para escalonar a instância $(n_1, \ldots, n_{k^2})$.

Recorrência para $\mathrm{OPT}$:

$$\mathrm{OPT}(n_1, \ldots, n_{k^2}) = 1 + \min_{(s_1,\ldots,s_{k^2}) \in \mathcal{C}} \mathrm{OPT}(n_1 - s_1, \ldots, n_{k^2} - s_{k^2})$$

# PD para as tarefas longas

$\mathrm{opt}(n_1, \ldots, n_{k^2})$: número mínimo de máquinas suficiente para escalonar a instância $(n_1, \ldots, n_{k^2})$.

Recorrência para $\mathrm{OPT}$:

$$\mathrm{OPT}(n_1, \ldots, n_{k^2}) = 1 + \min_{(s_1,\ldots,s_{k^2}) \in \mathcal{C}} \mathrm{OPT}(n_1 - s_1, \ldots, n_{k^2} - s_{k^2})$$

Note que $\mathrm{OPT} \leq m$ sse o $T$-escalonamento existe.

# PD para as tarefas longas

$\mathrm{opt}(n_1, \ldots, n_{k^2})$: número mínimo de máquinas suficiente para escalonar a instância $(n_1, \ldots, n_{k^2})$.

Recorrência para $\mathrm{OPT}$:

$$\mathrm{OPT}(n_1, \ldots, n_{k^2}) = 1 + \min_{(s_1,\ldots,s_{k^2}) \in \mathcal{C}} \mathrm{OPT}(n_1 - s_1, \ldots, n_{k^2} - s_{k^2})$$

Note que $\mathrm{OPT} \leq m$ sse o $T$-escalonamento existe.

Tabela com $n^{k^2}$ entradas.

# PD para as tarefas longas

$\mathrm{opt}(n_1, \ldots, n_{k^2})$: número mínimo de máquinas suficiente para escalonar a instância $(n_1, \ldots, n_{k^2})$.

Recorrência para $\mathrm{OPT}$:

$$\mathrm{OPT}(n_1, \ldots, n_{k^2}) = 1 + \min_{(s_1,\ldots,s_{k^2}) \in \mathcal{C}} \mathrm{OPT}(n_1 - s_1, \ldots, n_{k^2} - s_{k^2})$$

Note que $\mathrm{OPT} \leq m$ sse o $T$-escalonamento existe.

Tabela com $n^{k^2}$ entradas.

Cada entrada é calculada em tempo constante, pois $|\mathcal{C}|$ é constante.

# Como obter PTAS de $B_k$?

Fixe $\epsilon > 0$.

# Como obter PTAS de $B_k$?

Fixe $\epsilon > 0$.

Busca binária por um valor adequado de $T$.

# Como obter PTAS de $B_k$?

Fixe $\epsilon > 0$.

Busca binária por um valor adequado de $T$.

Durante o algoritmo, teremos um intervalo $[L, U]$ tal que:

- $C^*_{\max} \geq L$ e
- existe um $(1+\epsilon)U$-escalonamento.

# Como obter PTAS de $B_k$?

Fixe $\epsilon > 0$.

Busca binária por um valor adequado de $T$.

Durante o algoritmo, teremos um intervalo $[L, U]$ tal que:

- $C^*_{\max} \geq L$ e
- existe um $(1+\epsilon)U$-escalonamento.

Comece com

$$L_0 = \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\} \text{ e } U_0 = \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j.$$

Sabemos que $C^*_{\max} \in [L_0, U_0]$ e
o algoritmo de Graham produz um $U_0$-escalonamento.

# Como obter PTAS de $B_k$?

Seja $k = \lceil 1/\epsilon \rceil$.

Cada iteração começa com um intervalo $[L, U]$ tal que:

- $C^*_{\max} \geq L$ e
- existe um $(1+\epsilon)U$-escalonamento.

# Como obter PTAS de $B_k$?

Seja $k = \lceil 1/\epsilon \rceil$.

Cada iteração começa com um intervalo $[L, U]$ tal que:

- $C^*_{\max} \geq L$ e
- existe um $(1+\epsilon)U$-escalonamento.

Enquanto $L < U$ faça

Calcule $T = \lfloor \frac{L+U}{2} \rfloor$ e execute $B_k$ com esse $T$.

# Como obter PTAS de $B_k$?

Seja $k = \lceil 1/\epsilon \rceil$.

Cada iteração começa com um intervalo $[L, U]$ tal que:

- $C^*_{\max} \geq L$ e
- existe um $(1+\epsilon)U$-escalonamento.

Enquanto $L < U$ faça

Calcule $T = \lfloor \frac{L+U}{2} \rfloor$ e execute $B_k$ com esse $T$.

Se $B_k$ produz um escalonamento,
então $U \leftarrow T$
senão $L \leftarrow T + 1$

# Como obter PTAS de $B_k$?

Seja $k = \lceil 1/\epsilon \rceil$.

Cada iteração começa com um intervalo $[L, U]$ tal que:

- $C^*_{\max} \geq L$ e
- existe um $(1+\epsilon)U$-escalonamento.

Enquanto $L < U$ faça

Calcule $T = \lfloor \frac{L+U}{2} \rfloor$ e execute $B_k$ com esse $T$.

Se $B_k$ produz um escalonamento,
então $U \leftarrow T$
senão $L \leftarrow T + 1$

Devolva o $(1+\epsilon)U$-escalonamento.

# PTAS para o problema

$M$: vetor com lista de tarefas escalonadas a cada máquina.

Algoritmo $A_\epsilon\ (m, n, d)$

1 $k \leftarrow \lceil 1/\epsilon \rceil$

2 $L \leftarrow \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\}$

3 $U \leftarrow \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j$

4 $M \leftarrow$ ESCALONAMENTO-GRAHAM$(m, n, d)$

5 **enquanto** $L < U$ **faça**

6 $\quad T \leftarrow \lfloor \frac{L+U}{2} \rfloor$

7 $\quad M \leftarrow B_k(m, n, d, T)$

8 $\quad$ **se** $M$ é um escalonamento

9 $\quad\quad$ **então** $U \leftarrow T$

10 $\quad\quad$ **senão** $L \leftarrow T + 1$

11 **devolva** $M$

# PTAS para o problema

Algoritmo $A_\epsilon\ (m, n, d)$

1 $k \leftarrow \lceil 1/\epsilon \rceil$

2 $L \leftarrow \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\}$

3 $U \leftarrow \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j$

4 $M \leftarrow$ ESCALONAMENTO-GRAHAM$(m, n, d)$

5 **enquanto** $L < U$ **faça**

6 $\quad T \leftarrow \lfloor \frac{L+U}{2} \rfloor$

7 $\quad M \leftarrow B_k(m, n, d, T)$

8 $\quad$ **se** $M$ é um escalonamento

9 $\quad\quad$ **então** $U \leftarrow T$

10 $\quad\quad$ **senão** $L \leftarrow T + 1$

11 **devolva** $M$

# PTAS para o problema

Algoritmo $A_\epsilon\ (m, n, d)$

1 $k \leftarrow \lceil 1/\epsilon \rceil$

2 $L \leftarrow \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\}$

3 $U \leftarrow \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j$

4 $M \leftarrow$ ESCALONAMENTO-GRAHAM$(m, n, d)$

5 **enquanto** $L < U$ **faça**

6 $\quad T \leftarrow \lfloor \frac{L+U}{2} \rfloor$

7 $\quad M \leftarrow B_k(m, n, d, T)$

8 $\quad$ **se** $M$ é um escalonamento

9 $\quad\quad$ **então** $U \leftarrow T$

10 $\quad\quad$ **senão** $L \leftarrow T + 1$

11 **devolva** $M$

Correção: Ao final, sabemos que $C^*_{\max} \geq L$ e temos um $(1 + \epsilon)L$-escalonamento.

# Consumo de tempo

Algoritmo $A_\epsilon\ (m, n, d)$

1 $k \leftarrow \lceil 1/\epsilon \rceil$

2 $L \leftarrow \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\}$

3 $U \leftarrow \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j$

4 $M \leftarrow$ ESCALONAMENTO-GRAHAM$(m, n, d)$

5 **enquanto** $L < U$ **faça**

6 $\quad T \leftarrow \lfloor \frac{L+U}{2} \rfloor$

7 $\quad M \leftarrow B_k(m, n, d, T)$

8 $\quad$ **se** $M$ é um escalonamento

9 $\quad\quad$ **então** $U \leftarrow T$

10 $\quad\quad$ **senão** $L \leftarrow T + 1$

11 **devolva** $M$

# Consumo de tempo

Algoritmo $A_\epsilon\ (m, n, d)$

1 $k \leftarrow \lceil 1/\epsilon \rceil$
2 $L \leftarrow \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\}$
3 $U \leftarrow \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j$
4 $M \leftarrow$ ESCALONAMENTO-GRAHAM$(m, n, d)$
5 **enquanto** $L < U$ **faça**
6 $\quad T \leftarrow \lfloor \frac{L+U}{2} \rfloor$
7 $\quad M \leftarrow B_k(m, n, d, T)$
8 $\quad$ **se** $M$ é um escalonamento
9 $\quad\quad$ **então** $U \leftarrow T$
10 $\quad\quad$ **senão** $L \leftarrow T + 1$
11 **devolva** $M$

Número máximo de iterações: $\mathrm{O}(\lg(\max_j d_j))$.

# Consumo de tempo

Algoritmo $A_\epsilon\ (m, n, d)$

1 $k \leftarrow \lceil 1/\epsilon \rceil$
2 $L \leftarrow \max\{\lceil \frac{\sum_j d_j}{m} \rceil, \max_j d_j\}$
3 $U \leftarrow \lceil \frac{\sum_j d_j}{m} \rceil + \max_j d_j$
4 $M \leftarrow$ ESCALONAMENTO-GRAHAM$(m, n, d)$
5 **enquanto** $L < U$ **faça**
6 $\quad T \leftarrow \lfloor \frac{L+U}{2} \rfloor$
7 $\quad M \leftarrow B_k(m, n, d, T)$
8 $\quad$ **se** $M$ é um escalonamento
9 $\quad\quad$ **então** $U \leftarrow T$
10 $\quad\quad$ **senão** $L \leftarrow T + 1$
11 **devolva** $M$

Número máximo de iterações: $\mathrm{O}(\lg(\max_j d_j))$.

Cada iteração é polinomial em $m$ e $n$, mas exponencial em $k$.

# FPTAS para esse problema?

Há um PTAS para escalonamento em máquinas idênticas.

Será que existe um FPTAS?

# FPTAS para esse problema?

Há um PTAS para escalonamento em máquinas idênticas.

Será que existe um FPTAS?

problema é fortemente NP-difícil:

Qualquer que seja o polinômio $q(n)$,
o problema restrito às instâncias em que $d_i \leq q(n)$ para todo $i$ ainda é NP-difícil, onde $n$ é o número de tarefas.

# FPTAS para esse problema?

Há um PTAS para escalonamento em máquinas idênticas.

Será que existe um FPTAS?

problema é fortemente NP-difícil:

Qualquer que seja o polinômio $q(n)$,
o problema restrito às instâncias em que $d_i \leq q(n)$ para todo $i$ ainda é NP-difícil, onde $n$ é o número de tarefas.

Suponha que existe um FPTAS para o problema.

Vamos descrever um algoritmo polinomial que resolve o problema restrito às instâncias com $d_i \leq q(n)$ para todo $i$.