Problema do Escalonamento

- **Instância:** n tarefas com tempos $p_1, \ldots, p_n$ e m máquinas idênticas.
- **Objetivo:** Escalonamento (tarefas em máquinas) com tempo min.
- **Outros nomes:** Min makespan scheduling, multiprocessor sch, Job shop sch, identical parallel machine sch.
- **Tempo mínimo** $opt(m, n, p) \geq LB = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, \max_{i=1}^{n}\{p_i\}\right\}$

Relação com outros problemas

- **Bin Packing:** empacotar todos os itens em caixas de capacidade C. Minimizar o número de caixas.
- **Escalonamento:** empacotar todos os itens em m caixas. Minimizar a capacidade C das caixas.
- **Mochila:** empacotar alguns itens em apenas uma caixa com capacidade C. Maximizar o valor obtido.

Problema do Escalonamento

- **Instância:** n tarefas com tempos $p_1, \ldots, p_n$ e m máquinas idênticas.
- **Objetivo:** Escalonamento (tarefas em máquinas) com tempo min.
- **Outros nomes:** Min makespan scheduling, multiprocessor sch, Job shop sch, identical parallel machine sch.
- **Tempo mínimo** $opt(m, n, p) \geq LB = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, \max_{i=1}^{n}\{p_i\}\right\}$

Algoritmo Escalonamento-Graham(m, n, p)

1. **para** $i = 1$ até n **faça**
2. Atribua a tarefa i à máquina com menor tempo de trabalho

Algoritmo Escalonamento-Graham é 2-aproximativo

- Seja k a tarefa que termina por último no escalonamento
- Tempo final: $inicio_k$ (tempo de início da tarefa k) $+ p_k$
- $inicio_k$ foi selecionado para a máquina menos ocupada: $inicio_k \leq \frac{1}{m}\sum_{i=1}^{n} p_i \leq LB \leq opt(m, n, p)$
- Tempo final $= inicio_k + p_k \leq LB + LB = 2 \cdot LB \leq 2 \cdot opt(m, n, p)$

Problema do Escalonamento - Algoritmo 1.5-aprox

Algoritmo Esc-Graham-Decreasing(m, n, p)

1. **Ordene** as tarefas da maior para a menor
2. **para** $i = 1$ até n **faça**
3. Atribua a tarefa i à máquina com menor tempo de trabalho

Algoritmo Esc-Graham-Decreasing é (3/2)-aproximativo

- Podemos assumir que a tarefa n é a última a terminar no escalon
- (*) CC, a inclusão da tarefa n não muda o tempo do escalonamento, mas aumenta o tempo ótimo, em comparação com a instância sem a tarefa n. Ou seja, o fator de aproximação diminui
- Tempo final: $inicio_n$ (tempo de início da tarefa n) $+ p_n$
- $inicio_n$ foi selecionado pra máquina menos ocupada: $inicio_n \leq \frac{1}{m} \sum_{i=1}^{n} p_i \leq LB \leq opt$
- Tempo final $= inicio_n + p_n \leq opt + p_n$
- $p_n \leq opt/2 \Rightarrow$ Tempo final $\leq (3/2) \cdot opt(m, n, p)$. **OK**
- $p_n > opt/2 \Rightarrow opt < 2 \cdot p_n \Rightarrow$ só 1 tarefa por máquina no Opt $\Rightarrow$ Esc-Graham-Decreasing retorna o ótimo

Problema do Escalonamento - Algoritmo 1.333-aprox

Algoritmo Esc-Graham-Decreasing é (4/3)-aproximativo

- Tempo final: $inicio_n$ (tempo de início da tarefa n) + p_n
- $inicio_n$ foi selecionado pra máquina menos ocupada: $inicio_n \leq \frac{1}{m}\sum_{i=1}^{n} p_i \leq LB \leq opt$
- Tempo final $= inicio_n + p_n \leq opt + p_n$
- $p_n \leq opt/3 \Rightarrow$ Tempo final $\leq (4/3) \cdot opt(m, n, p)$. **OK**
- Assuma $p_n > opt/3$: $opt < 3 \cdot p_n \Rightarrow$ Máx. 2 tarefas por máquina.
- Provar que nesse caso Esc-Graham-Decreasing retorna o ótimo (opt)
- Tarefa maior em Maq_1, 2^o maior em Maq_2,..., m^o maior em Maq_m.
- Depois atribui as $n - m$ restantes ($n \leq 2m$) em Maq_m, Maq_{m-1},...
- Suponha que Maq_k passou tempo opt (2 tarefas: grd e peq)
- Trocar peq para Máq com só 1 tarefa ($\geq$ grd): $> opt$
- Trocar peq por tarefa menor, peq ficará com tarefa maior: $> opt$
- Trocar peq por tarefa maior, esta ficará com tarefa grd: $> opt$
- Resumindo: é impossível escalonar sem passar opt. Contradição.

Problema do Escalonamento - Exemplos tight

Escalonamento-Graham é 2-aprox (e não menos !!)

- $m(m-1)$ tarefas de tempo 1 e 1 tarefa de tempo m
- $opt = m$ (1 máquina com a de tempo m e as $m-1$ restantes com m tarefas de tempo 1 cada)
- Escalonamento-Graham coloca $m-1$ tarefas de tempo 1 em cada máquina e depois acrescenta a de tempo m em uma delas, gerando tempo total $(m-1)+m = 2m-1$
- Fator de aproximação: $(2m-1)/m = 2 - 1/m$

Esc-Graham-Decreasing é 4/3-aprox (e não menos !!)

- 3 tarefas de tempo m e 2 tarefas de tempos $m+1, m+2, \ldots, 2m-1$
- $opt = 3m$ (1 máquina com as 3 de tempo m e as $m-1$ restantes com 2 tarefas de tempos $m+i$ e $2m-i$)
- Esc-Graham-Decreasing põe $2m$ tarefas primeiro (2 em cada máq): tempo total $3m^2 - m = m(3m-1)$. Depois põe a última (tempo m)
- Fator de aproximação: $(4m-1)/(3m) = 4/3 - 1/(3m)$

PTAS p/ Escalonamento com núm m fixo de máquinas

- Fixe ε. Uma tarefa é pequena se $\leq \varepsilon \cdot LB_2 = \frac{\varepsilon}{m} \sum_{i=1}^{n} p_i$
- Há no máximo m/ε tarefas grandes.
- Tempo $O(m^{m/\varepsilon})$: escalonamento opt para tarefas grandes
- Escalonamento-Graham nas tarefas pequenas
- Se a última tarefa é grande, **OK**
- Senão, seja k a última tarefa (pequena) do escalonamento
- Tempo final: $inicio_k$ (tempo de início da tarefa k) $+ p_k$
- $inicio_k$ foi selecionado para a máquina menos ocupada:
  $inicio_k \leq \frac{1}{m} \sum_{i=1}^{n} p_i = LB_2 \leq opt(m, n, p)$
- Tempo final do escalonamento $=$
  $inicio_k + p_k \leq LB_2 + \varepsilon \cdot LB_2 = (1+\varepsilon) \cdot LB_2 \leq (1+\varepsilon) \cdot opt$
- Tempo do algoritmo $O(m^{m/\varepsilon} + m \cdot n)$

PTAS p/ Escalonamento com núm qualquer de máquinas

- **Instância:** n tarefas com tempos $\mathbf{p} = (p_1, \ldots, p_n)$ e m máq. idênt.
- $\exists$ escalonamento com tempo total t **se e só se** $\exists$ empacotamento de itens com pesos $\mathbf{p} = (p_1, \ldots, p_n)$ em m caixas de capacidade t
- $bins(\mathbf{p}, t)$: menor número de caixas de capacidade t para empacotar itens com pesos $\mathbf{p}$. $\quad bins(\mathbf{p}, t) = bins(\mathbf{p}/t, 1)$
- Tempo min Escalonamento **opt** $= \min\{t : bins(\mathbf{p}, t) \leq m\}$
- PTAS assintótico para Bin Packing $\implies$ PTAS para Escalonamento

Relação com outros problemas

- **Bin Packing:** empacotar todos os itens em caixas de capacidade t. Minimizar o número de caixas.
- **Escalonamento:** empacotar todos os itens em m caixas. Minimizar a capacidade t das caixas.
- **Mochila:** empacotar alguns itens em apenas uma caixa com capacidade t. Maximizar o valor obtido.

PTAS para o Problema do Escalonamento

- **Instância:** n tarefas com tempos $\mathbf{p} = (p_1, \ldots, p_n)$ e m máq. idênt.
- Seja $t \in [LB, 2 \cdot LB]$. $p_j \leq t, \forall j$. Ignorar itens pequenos* $(\leq t\varepsilon)$
- $p_j \in [t\varepsilon(1+\varepsilon)^i, t\varepsilon(1+\varepsilon)^{i+1}) \Rightarrow p'_j = t\varepsilon(1+\varepsilon)^i$.
- $\varepsilon(1+\varepsilon)^K = 1 \Rightarrow K = \lceil \log_{1+\varepsilon}(1/\varepsilon) \rceil$ valores diferentes em $\mathbf{p}'$
- Programação Dinâmica $bins(\mathbf{p}', t)$ em tempo $O(n^{2K})$.
- Reduz por fator $1+\varepsilon \Rightarrow bins^*(\mathbf{p}, t(1+\varepsilon)) \leq bins^*(\mathbf{p}', t)$
- Empacotar itens pequenos $(\leq t\varepsilon)$ com FF. Seja $bins^{FF}(\mathbf{p}, t(1+\varepsilon))$ o retornado por esse procedimento. Se não abriram caixas para peq.:
- $bins^{FF}(\mathbf{p}, t(1+\varepsilon)) = bins^*(\mathbf{p}, t(1+\varepsilon)) \leq bins^*(\mathbf{p}', t) \leq bins(\mathbf{p}, t)$
- CC: todas caixas (exceto ≤ 1) estão cheias até $\geq t$. Então o ótimo deve usar pelo menos a mesma quantidade de caixas.
- **Resumindo:** $bins^{FF}(\mathbf{p}, t(1+\varepsilon)) \leq bins(\mathbf{p}, t)$
- $\mathbf{opt} = \min\{t : bins(\mathbf{p}, t) \leq m\} \geq \min\{t : bins^{FF}(\mathbf{p}, t(1+\varepsilon)) \leq m\}$
- Busca binária por $T \in [LB, 2 \cdot LB]$ até subintervalo de tam. $\varepsilon \cdot LB$

PTAS para o Problema do Escalonamento

- $\mathbf{opt} = \min\{t : \mathit{bins}(\mathbf{p}, t) \leq m\} \geq \min\{t : \mathit{bins}^{FF}(\mathbf{p}, t(1+\varepsilon)) \leq m\}$
- Busca binária por $t \in [LB, 2 \cdot LB]$ até subintervalo de tam. $\varepsilon \cdot LB$
- $\frac{1}{\varepsilon}$ subint. desse tam. $\Rightarrow$ núm testes da busca binária: $\leq \lceil \log_2(\frac{1}{\varepsilon}) \rceil$.
- Seja T o extremo direito do subintervalo: $T \leq (1+\varepsilon) \cdot \mathbf{opt}$, pois
- $T \leq \min\{t : \mathit{bins}^{FF}(\mathbf{p}, t(1+\varepsilon)) \leq m\} + \varepsilon \cdot LB \leq \mathbf{opt} + \varepsilon \cdot \mathbf{opt}$
- Finalmente: o algoritmo retorna tempo $T(1+\varepsilon) \leq (1+\varepsilon)^2 \cdot \mathbf{opt}$
- Algoritmo $(1+3\varepsilon)$-aprox. em tempo $O(n^{2K} \log_2(\frac{1}{\varepsilon}))$, onde $K = \lceil \log_{1+\varepsilon}(\frac{1}{\varepsilon}) \rceil$. PTAS

FPTAS p/ Escalonamento com 2 máquinas

- Para simplificar, vamos assumir que os tempos $p_1, \ldots, p_n$ são inteiros
- Seja $P = \sum_{i=1}^{n} p_i$. Então: $\log_2 P \leq \langle \mathrm{I} \rangle \leq n \log_2 P$.
- Par (x, y) significa que M_1 e M_2 estão com tempos x e y, resp.

Algoritmo Escalonamento-2-Máquinas($\mathbf{p}, n$)

1. $VS_0 \leftarrow \{(0,0)\}$
2. **para** $k \leftarrow 1$ **até** n **faça**
3. $\quad VS_k \leftarrow \emptyset$
4. $\quad$ **para** cada par $(x, y) \in VS_{k-1}$ **faça**
5. $\quad\quad VS_k \leftarrow VS_k \cup \{(x + p_k, y),\ (x, y + p_k)\}$
6. **retorne** o par $(x, y) \in VS_n$ com menor $\max\{x, y\}$

- $(x, y) \in VS_k \Rightarrow 0 \leq x, y \leq P \Rightarrow$ Tempo $O(n \cdot P^2)$
- Esse tempo pode ser exponencial no tamanho $\langle \mathrm{I} \rangle$ da instância I.
- Truque já usado: Dividir valores $\{0, \ldots, P\}$ em L grupos.
- $L = O((1/\varepsilon) \cdot n \ln P) \Rightarrow$ Tempo $O(n \cdot L^2) = \boxed{O((1/\varepsilon^2) \cdot n^3 \cdot (\ln P)^2)}$

FPTAS p/ Escalonamento com 2 máquinas

- Truque já usado: Dividir valores $\{0, \ldots, P\}$ em L grupos.
- $L = O((1/\varepsilon) \cdot n \ln P) \Rightarrow$ Tempo $O(n \cdot L^2) = \boxed{O((1/\varepsilon^2) \cdot n^3 \cdot (\ln P)^2)}$
- Grupos formados com cortes nos pontos λ^i, onde $\lambda = 1 + \frac{\varepsilon}{2n}$
- $[0, \lambda),\ [\lambda, \lambda^2),\ [\lambda^2, \lambda^3),\ \ldots,\ [\lambda^{L-1}, \lambda^L]$. Valores em **p** são inteiros
- $L = \lceil \log_\lambda P \rceil \leq \lceil (1 + \frac{2n}{\varepsilon}) \ln P \rceil$, pois $\ln z \geq (z-1)/z$ para $z \geq 1$
- x_1, x_2 no mesmo grupo $\Rightarrow x_1/\lambda \leq x_2 \leq x_1 \cdot \lambda$
- Montar L^2 grupos de $\{0, \ldots, P\}^2$. Montar $VS_k^\#$ a partir de VS_k, deixando no máximo um elemento em cada grupo de pares. Na verdade, $VS_k^\#$ é montado a partir de $VS_{k-1}^\#$ seguindo o algoritmo.

- Provar que, se $(x, y) \in VS_k$, então existe $(x^\#, y^\#) \in VS_k^\#$ tal que $x^\# \leq \lambda^k x$ e $y^\# \leq \lambda^k y$
- **Com isso:** Se $(x, y) \in VS_n$ é o par ótimo, então $\exists (x^\#, y^\#) \in VS_n^\#$ tal que: $\max\{x^\#, y^\#\} \leq \lambda^n \max\{x, y\} = \lambda^n \cdot \mathbf{opt}$,
- **Fator de aproximação** $\lambda^n = (1 + \frac{\varepsilon}{2n})^n \leq (1 + \varepsilon)$. **FPTAS**
- Pois $(1 + z/n)^n \leq 1 + 2z$, para $z \in [0, 1]$

FPTAS p/ Escalonamento com 2 máquinas

LEMA: Se $(x, y) \in VS_k$, então existe $(x^\#, y^\#) \in VS_k^\#$ tal que $x^\# \leq \lambda^k x$ e $y^\# \leq \lambda^k y$. Ou seja, $VS_k^\#$ é uma aproximação decente de VS_k.

- Indução em k. Para $k = 1$: já visto. Suponha que vale para $k - 1$.
- $(x, y) \in VS_k \Rightarrow (x, y)$ é $(a + p_k, b)$ ou $(a, b + p_k)$ p/ $(a, b) \in VS_{k-1}$
- Por indução, $\exists (a^\#, b^\#) \in VS_{k-1}^\#$: $a^\# \leq \lambda^{k-1} a$ e $b^\# \leq \lambda^{k-1} b$
- Na iteração k, $(a^\# + p_k, b^\#)$ é gerado, mas pode ser descartado por um par (α, β) do mesmo grupo ao se montar $VS_k^\#$
- $\alpha \leq \lambda(a^\# + p_k) \leq \lambda^k a + \lambda p_k \leq \lambda^k (a + p_k) = \lambda^k x$
- $\beta \leq \lambda b^\# \leq \lambda^k b = \lambda^k y$ $\square$

FPTAS p/ Escalonamento com m máquinas (m fixo)

Generalização para m máquinas

- Mesmo algoritmo, mas m-uplas $(x_1, \ldots, x_m)$ ao invés de pares (x, y)
- Tempo $O(n \cdot P^m)$.
- Mesma partição em L grupos, com $L = O((1/\varepsilon) \cdot n \ln P)$
- Dividir m-uplas em L^m grupos
- FPTAS com tempo $O(n \cdot L^m) = \boxed{O((1/\varepsilon)^m \cdot n^{m+1} \cdot (\ln P)^m)}$
- com mesmo fator de aproximação $1 + \varepsilon$

Escalonamento versus Escalonamento-Int, com m fixo

- Já vimos que os problemas Escalonamento e Escalonamento-Int são polinomialmente equivalentes. Pode-se obter um FPTAS para um a partir de um FPTAS do outro.
- Escalonamento com m fixo tem FPTAS

Algoritmo Aproximativo Probabilístico p/ Escalonamento

Algoritmo Esc-Graham-Aleatório(m, n, p)

1. **ordene** as tarefas de modo aleatório
2. **para** $i = 1$ até n **faça**
3. Atribua a tarefa i à máquina com menor tempo de trabalho

Algoritmo Esc-Graham-Aleatório é 2-aprox (e não menos !!)

- **Instância limite:** n tarefas tempo n e $n^2(n+1)$ tarefas tempo 1 em $n(n+2)$ máquinas. **Ótimo:** tempo $opt = n$ em cada máquina.
- **Ordem aleatória:** Sorteia unif. número em $[0, 1]$ para cada tarefa. A ordem desses números determina a ordem das tarefas.
- **Order Statistics:** Número última tarefa **grande** $\approx \frac{n}{n+1} = 1 - \frac{1}{n+1}$
- **Com alta prob:** Núm tarefas **peq.** antes: $\geq \left(1 - \frac{c+1}{n+1}\right) \cdot n^2(n+1)$
- **Graham:** tempo $\geq n + \frac{n^2(n-c)}{n(n+2)} \geq \left(2 - \frac{c+2}{n+2}\right) n$
- $n \to \infty$, $c = o(n)$: **Razão** $2 - o(1) \to$ **2**

Escalonamento $\times$ 3-Partition

Problema 3-Partition de decisão (3 tarefas por máquina)

- **Instância:** $n = 3m$ tarefas com tempos $p_1, \ldots, p_n$ e m máquinas idênticas. Ademais, $LB_2/4 < p_k < LB_2/2$, onde $LB_2 = \frac{1}{m}\sum_{i=1}^{n} p_i$
- **Decisão:** Existe escalonam com tempo$= LB_2$ em cada máquina?
- **Obs:** A restrição das instâncias obrigam 3 tarefas por máquina.
- **Otimização:** Minimizar tempo total.

Observações

- 3-Partition é um problema de decisão clássico
- 3-Partition é NP-Completo Forte (Livro de Garey & Johnson)
- A versão opt de 3-Partition é um caso particular de Escalonamento
- Escalonamento Mínimo é NP-Difícil Forte

Conclusão (até agora)

* **Escalonamento:** PTAS (p/ m qquer) e FPTAS (p/ m fixo*).
* **TSPM:** Alg 1.5-aproximativo. Parece ser o melhor possível, na prática.
* **Weighted Set Cover:** Algoritmo $(\ln n+1)$-aprox. Parece o melhor possível (parece não ter aprox para fator constante).
* **Set Cover:** Algoritmos $(\ln n+1)$-aprox. e f-aprox. Parecem os melhores possíveis (parece não ter aprox para fator constante).
* **Vertex Cover:** Alg 2-aprox. Parece o melhor possível.
* **Mochila:** FPTAS. É o melhor possível.
* **Bin Packing:** PTAS assintótico (parece não ter FPTAS assintótico).

- Problemas NP-Difíceis com FPTAS (Mochila, Escalonamento*)
- Problemas NP-Difíceis com PTAS, prov **sem** FPTAS (Escalonam)
- Problemas NP-Difíceis com α-aprox (p/ α const), provav **sem** PTAS (TSP Métrico, Vertex Cover, Bin Packing)
- Problemas NP-Difíceis com $\alpha(n)$-aprox (p/ $\alpha(n)$ **não** const), que provav **não** tem para α const (Set Cover)
- Problemas NP-Difíceis que provav **não** tem alg poli $\alpha(n)$-aprox para nenhuma função computável $\alpha(n)$ poli (Caixeiro viajante)