计算机问题求解 – 论题4–11

- Bin-Packing 问题

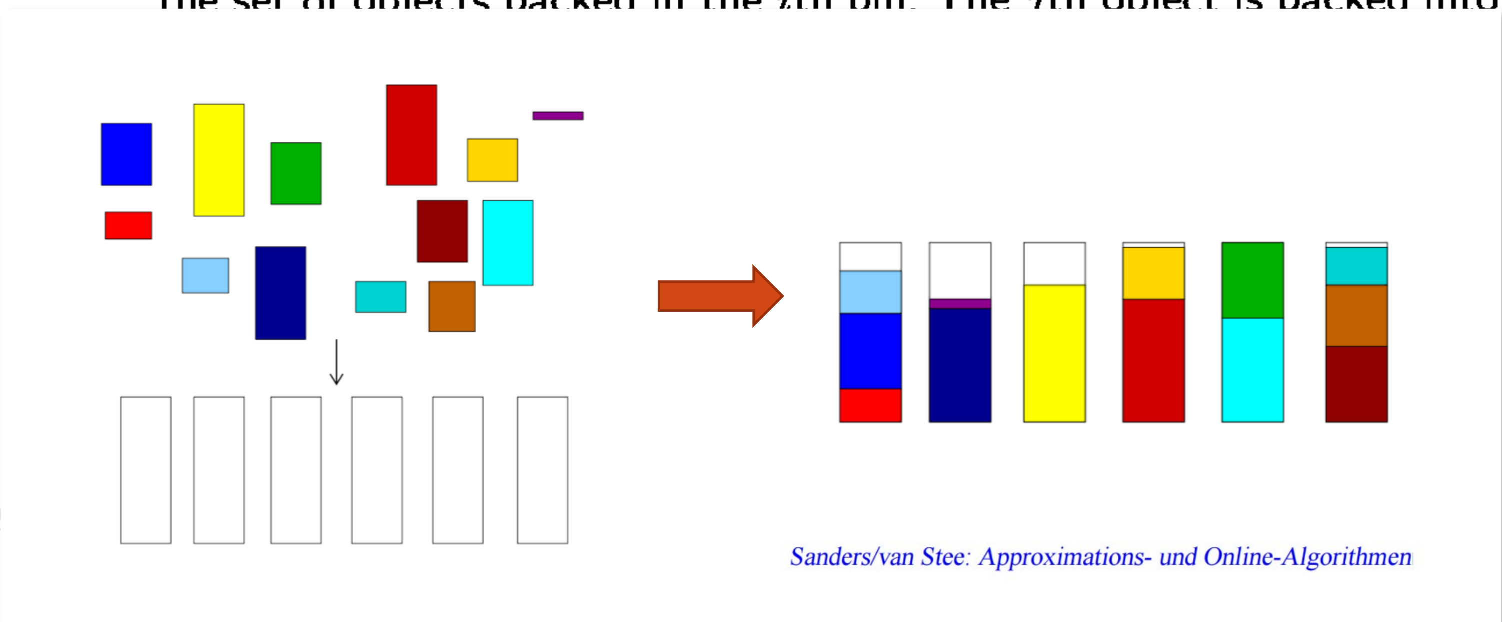
  课程研讨

  - JH第4章第3节第6小节

Bin-Packing 问题

Input: n rational numbers $w_1, w_2, \ldots, w_n \in [0,1]$ for some positive integer n.

Constraints: $\mathcal{M}(w_1, w_2, \ldots, w_n) = \{S \subseteq \{0,1\}^n \mid$ for every $s \in S$, $s^{\mathsf{T}} \cdot (w_1, w_2, \ldots, w_n) \leq 1$, and $\sum_{s \in S} s = (1, 1, \ldots, 1)\}$.
{If $S = \{s_1, s_2, \ldots, s_m\}$, then $s_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ determines the set of objects packed in the ith bin. The ith object is packed into



C

Goal: *minimum*.

两个不同的“balls”

Definition 4.2.3.1. *Let* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *and* $\overline{U} = (\Sigma_I, \Sigma_O, L, L, \mathcal{M}, cost, goal)$ *be two optimization problems with* $L_I \subset L$. *A* **distance function for** $\overline{U}$ **according to** L_I *is any function* $h_L : L \to \mathbb{R}^{\geq 0}$ *satisfying the properties*

(i) $h_L(x) = 0$ *for every* $x \in L_I$, *and*
(ii) h *is polynomial-time computable.*

Let h *be a distance function for* $\overline{U}$ *according to* L_I. *We define, for any* $r \in \mathbb{R}^+$,

相对于一个kernel（子问题）

$$\boxed{\boldsymbol{Ball_{r,h}(L_I)}} = \{w \in L \mid h(w) \leq r\}.$$[6]

Definition 4.2.4.1. *Let* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *be an optimization problem. A* **constraint distance function** *for* U *is any function* $h : L_I \times \Sigma_O^* \to \mathbb{R}^{\geq 0}$ *such that*

(i) $h(x, S) = 0$ *for every* $S \in \mathcal{M}(x)$,
(ii) $h(x, S) > 0$ *for every* $S \notin \mathcal{M}(x)$, *and*
(iii) h *is polynomial-time computable.*

相对于可行解的限制条件

For every $\varepsilon \in \mathbb{R}^+$, *and every* $x \in L_I$, $\boldsymbol{\mathcal{M}_\varepsilon^h(x)} = \{S \in \Sigma_O^* \mid h(x, S) \leq \varepsilon\}$ *is the* $\boxed{\boldsymbol{\varepsilon}\textbf{-ball of } \boldsymbol{\mathcal{M}(x)}}$ **according to** $\boldsymbol{h}$.

Dual Approximation Algorithm

Definition 4.2.4.2. *Let* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$ *be an optimization problem, and let* h *be a constraint distance function for* U.

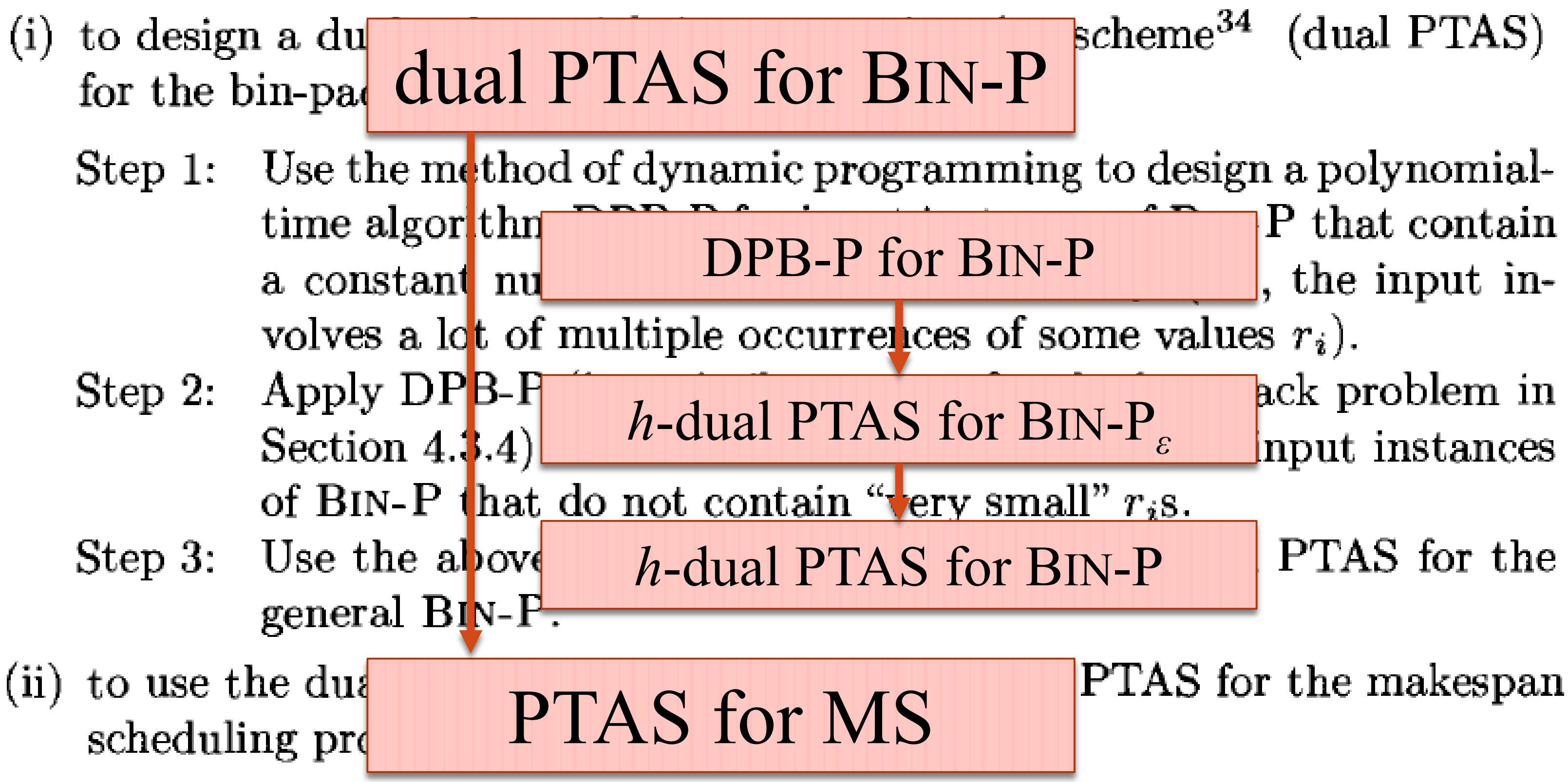
An optimization algorithm A *for* U *is called an* **h-dual ε-approximation algorithm for U**, *if for every* $x \in L_I$,

(i) $A(x) \in \mathcal{M}_\varepsilon^h(x)$, *and*
(ii) $cost(A(x)) \geq Opt_U(x)$ *if* $goal = maximum$, *and*
$cost(A(x)) \leq Opt_U(x)$ *if* $goal = minimum$.

- *h*-dual polynomial-time approximation scheme (***h*-dual PTAS** for *U*)
- *h*-dual fully polynomial-time approximation scheme (***h*-dual FPTAS** for *U*)

Dual Approximation Algorithms

- 这节最终目标是求解MS，你能解释整体思路吗？



dual PTAS for BIN-P (1)

Step 1: Use the method of dynamic programming to design a polynomial-time algorithm DPB-P for input instances of BIN-P that contain a constant number of different values of r_is (i.e., the input involves a lot of multiple occurrences of some values r_i).

- 你能解释动态规划的递归式吗？

$$\text{Bin-P}(m_1, \ldots, m_s) =$$

$$1 + \min_{x_1, \ldots, x_s} \left\{ \text{Bin-P}(m_1 - x_1, \ldots, m_s - x_s) \,\middle|\, \sum_{i=1}^{s} x_i q_i \le 1 \right\}.$$

dual PTAS for BIN-P (2)

- 你能解释算法4.3.6.1及其时间复杂度吗？

Algorithm 4.3.6.1 (DPB-P$_s$).

Input: $q_1, \ldots, q_s$, $n_1, \ldots, n_s$, where $q_i \in (0, 1]$ for $i = 1, \ldots, s$, and $n_1, \ldots, n_s$ are positive integers.

Step 1: $\text{BIN-P}(0, \ldots, 0) := 0$;
$\text{Bin-P}(h_1, \ldots, h_s) := 1$ for all $(h_1, \ldots, h_s) \in \{0, \ldots, n_1\} \times \cdots \times \{0, \ldots, n_s\}$ such that $\sum_{i=1}^{s} h_i q_i \leq 1$ and $\sum_{i=1}^{s} h_i \geq 1$.

Step 2: Compute $\text{Bin-P}(m_1, \ldots, m_s)$ with the corresponding optimal solution $T(m_1, \ldots, m_s)$ by the recurrence (4.61) for all $(m_1, \ldots, m_s) \in \{0, \ldots, n_1\} \times \cdots \times \{0, \ldots, n_s\}$.

Output: $\text{BIN-P}(n_1, \ldots, n_s)$, $T(m_1, \ldots, m_s)$.

$$\boxed{n_1 \cdot n_2 \cdot \cdots \cdot n_s \leq \left(\frac{\sum_{i=1}^{s} n_i}{s}\right)^s = \left(\frac{n}{s}\right)^s} \Rightarrow O\left(\left(\frac{n}{s}\right)^{2s}\right)$$

Bin-Packing 问题

Input: n rational numbers $w_1, w_2, \ldots, w_n \in [0,1]$ for some positive integer n.

Constraints: $\mathcal{M}(w_1, w_2, \ldots, w_n) = \{S \subseteq \{0,1\}^n \mid$ for every $s \in S$, $s^{\mathsf{T}} \cdot (w_1, w_2, \ldots, w_n) \leq 1$, and $\sum_{s \in S} s = (1, 1, \ldots, 1)\}$.
{If $S = \{s_1, s_2, \ldots, s_m\}$, then $s_i = (s_{i1}, s_{i2}, \ldots, s_{in})$ determines the set of objects packed in the ith bin. The jth object is packed into the ith bin if and only if $s_{ij} = 1$. The constraint

$$\boxed{s_i^{\mathsf{T}} \cdot (w_1, \ldots, w_n) \leq 1}$$

to be relaxed

assures that the ith bin is not overfilled. The constraint

$$\sum_{s \in S} s = (1, 1, \ldots, 1)$$

assures that every object is packed in exactly one bin.}

Cost: For every $S \in \mathcal{M}(w_1, w_2, \ldots, w_n)$,

$$cost(S, (w_1, \ldots, w_n)) = |S|.$$

Goal: *minimum*.

dual PTAS for BIN-P (3)

Step 2: Apply DPB-P (in a similar way as for the knapsack problem in Section 4.3.4) to obtain an h-dual PTAS for the input instances of BIN-P that do not contain "very small" r_is.

- 你能解释算法4.3.6.2吗？它对输入做了怎样的处理？
- 它为什么是Bin-P$_\varepsilon$的h-dual ε-近似算法？

Algorithm 4.3.6.2 (BP-PTA$_\varepsilon$).

Input: $(q_1, q_2, \ldots, q_n)$, where $\varepsilon < q_1 \leq$

Step 1: Set $s := \lceil \log_2(1/\varepsilon)/\varepsilon \rceil$;
$l_1 := \varepsilon$, and
$l_j := l_{j-1} \cdot (1 + \varepsilon)$ for $j = 2, 3,$
$l_{s+1} = 1$.
{This corresponds to the partitioning of the interval $(\varepsilon, 1]$ into s subintervals $(l_1, l_2], (l_2, l_3], \ldots, (l_s, l_{s+1}]$.}

Step 2: **for** $i = 1$ **to** s **do**
do begin $L_i := \{q_1, \ldots, q_n\} \cap (l_i, l_{i+1}]$;
$n_i := |L_i|$
end
{We consider that every value of L_i is rounded to the value l_i in what follows.}

Step 3: Apply DPB-P$_s$ on the input $(l_1, l_2, \ldots, l_s, n_1, n_2, \ldots, n_s)$.

将这些输入划分（转变）为s个区间。

将该区间的下界作为此区间中的所有值。

BP-PTA$_\varepsilon$是一个h-dual ε–近似算法

To prove that BP-PTA$_\varepsilon$ is an h-dual ε-approximation algorithm for Bin-P$_\varepsilon$, we have to prove that, for every input $I = (q_1, q_2, \ldots, q_n)$, $\varepsilon < q_1 \leq \cdots \leq q_n \leq 1$, the following two facts hold:

(i) $r = cost(T_1, \ldots, T_r) = \text{Bin-P}(n_1, \ldots, n_s) \leq Opt_{\text{Bin-P}}(I)$, where $(T_1, \ldots, T_r)$ is the optimal solution for the input $Round(I) = (l_1, \ldots, l_s, n_1, \ldots, n_s)$ com … indices of th …

BP-PTA$_\varepsilon$的解一定优于最优解

(ii) for every $j = 1, \ldots, r$, $\sum_{a \in T_j} q_a \leq 1 + \varepsilon$.

The fact (i) is obvious because $Round(I)$ can be considered as $(p_1, \ldots, p_n)$, where $p_i \leq q_i$ for every $i \in \{1, \ldots, n\}$.

Since DPB-P$_s(Round(I)) \leq Opt_{\text{Bin-P}}(I)$, we obtain

$$\text{Bin-P}(n_1, \ldots, n_s) = Opt_{\text{Bin-P}}(Round(I)) \leq Opt_{\text{Bin-P}}(I).$$

To prove (ii), consider an arbitrary set of indices $T \in \{T_1, T_2, \ldots, T_r\}$. Let $x_T = (x_1, \ldots, x_n)$ be the corresponding description of the set of indices assigned to this bin for $Round(I)$. We can bound $\sum_{j \in T} q_j$ as follows:

$$\sum_{j \in T} q_j \leq \sum_{i=1}^{s} x_i l_{i+1} = \sum_{i=1}^{s} x_i l_i + \sum_{i=1}^{s} x_i (l_{i+1} - l_i) \leq 1 + \sum_{i=1}^{s} x_i (l_{i+1} - l_i). \quad (4.62)$$

Since $l_i > \varepsilon$ for every $i \in \{1, \ldots, s\}$, the number of pieces in a bin is at most $\lfloor \frac{1}{\varepsilon} \rfloor$, i.e.,

$$\sum_{i=1}^{s} x_i \leq \left\lfloor \frac{1}{\varepsilon} \right\rfloor . \quad (4.63)$$

Let, for $i = 1, \ldots$ by values of size l_i. Obviously,

$$\sum_{j \in T} q_j \leq 1 + \varepsilon$$

(4.64)

for every $i \in \{1, 2$ obtain

$$\begin{aligned}
\sum_{j \in T} q_j &\underset{(4.62)}{\leq} 1 + \sum_{i=1}^{s} x_i (l_{i+1} - l_i) \\
&\underset{(4.64)}{\leq} 1 + \sum_{i=1}^{s} \frac{a_i}{l_i} (l_{i+1} - l_i) \\
&= 1 + \sum_{i=1}^{s} \left[a_i \cdot \frac{l_{i+1}}{l_i} - a_i \right] \\
&= 1 + \sum_{i=1}^{s} a_i \cdot \left(\frac{l_{i+1}}{l_i} - 1 \right) \\
&= 1 + \sum_{i=1}^{s} a_i \cdot \varepsilon = 1 + \varepsilon \cdot \sum_{i=1}^{s} a_i = 1 + \varepsilon
\end{aligned}$$

用BP-PTA$_\varepsilon$求解BIN-P还存在什么问题？

要甩掉ε

dual PTAS for BIN-P (4)

Step 3: Use the above h-dual PTAS to design an h-dual PTAS for the general BIN-P.

- 你能解释算法4.3.6.4吗？对输入做了怎样的处理？
- 它为什么是BIN-P的h-dual PTAS？

Algorithm 4.3.6.4 (Bin-PTAS).

Input: (I, ε), where $I = (q_1, q_2, \ldots, q_n)$, $0 \leq q_1 \leq q_2 \leq \cdots \leq q_n \leq 1$, $\varepsilon \in (0, 1)$.

Step 1: Find i such that $q_1 \leq q_2 \leq \ldots \leq q_i \leq \varepsilon \leq q_{i+1} \leq q_{i+2} \leq \cdots \leq q_n$.

Step 2: Apply BP-PTA$_\varepsilon$ on the input $(q_{i+1}, \ldots, q_n)$. Let $T = (T_1, \ldots, T_m)$ be the output BP-PTA$_\varepsilon(q_{i+1}, \ldots, q_n)$.

Step 3: For every i such that $\sum_{j \in T_i} q_j \leq 1$ pack one of the small pieces from $\{q_1, \ldots, q_i\}$ into T_i until $\sum_{j \in T_i} q_j > 1$ for all $j \in \{1, 2, \ldots, n\}$. If there are still some small pieces to be assigned, take a new bin and pack the pieces there until this bin is overfilled. Repeat this last step several times, if necessary.

Proof. First, we analyze the time complexity of Bin-PTAS. Step 1 can be executed in linear time. (If one needs to sort the input values, then it takes $O(n \log n)$ time.) Following Lemma 4.3.6.3 the application of BP-PTA$_\varepsilon$ on the input values larger than ε runs in time polynomial according to n. Step 3 can be implemented in linear time.

Now we have to prove that for every input (I, ε), $I = (q_1, \ldots, q_n)$, $\varepsilon \in (0, 1)$,

(i) $cost(\text{Bin-PTAS}(I, \varepsilon)) \leq Opt_{\text{Bin-P}}(I)$, and
(ii) every bin of Bin-PTAS(I, ε) has a size of at most $1 + \varepsilon$.

The condition (ii) is obviously fulfilled because BP-PTA$_\varepsilon$ is an h-dual ε-approximation algorithm, i.e., the bins of BP-PTA$_\varepsilon(q_{i+1}, \ldots, q_n)$ have a size of at most $1 + \varepsilon$. One can easily observe that the small pieces $q_1, \ldots, q_i$ are added to BP-PTA$_\varepsilon(q_{i+1}, \ldots, q_n)$ in Step 3 in such a way that no bin has a size greater than $1 + \varepsilon$.

To prove (i) we first observe that (Lemma 4.3.6.3)

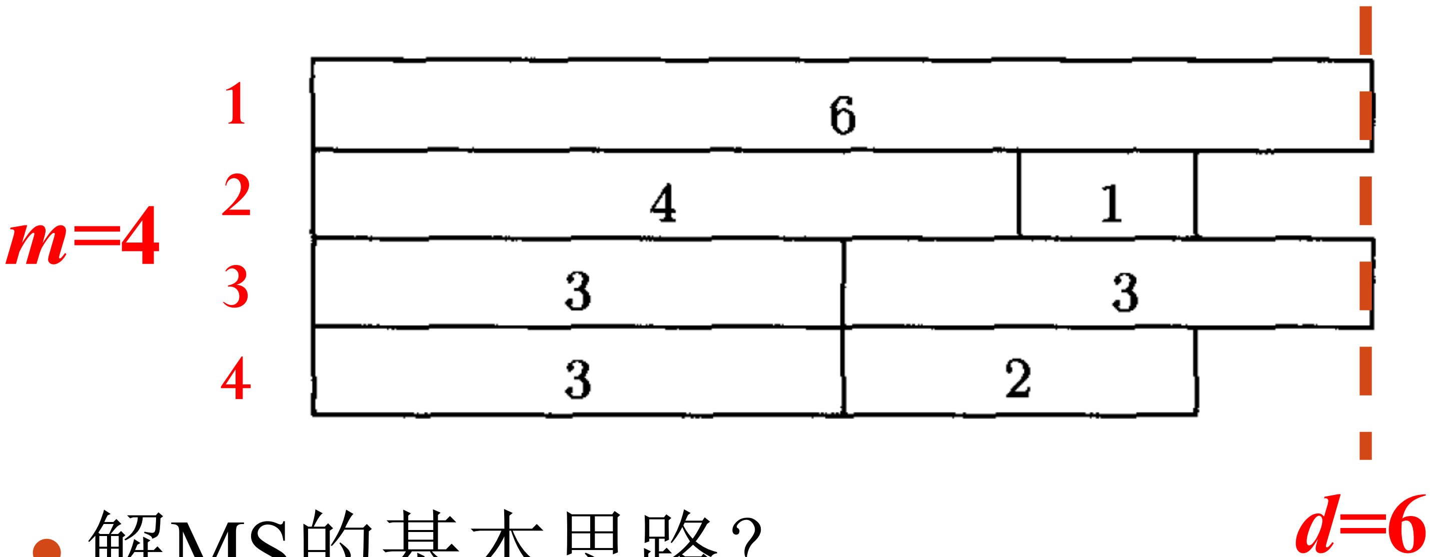
$$Opt_{\text{Bin-P}}(q_{i+1}, \ldots, q_n) \geq cost(\text{BP-PTA}_\varepsilon(q_{i+1}, \ldots, q_n)).$$

Now, if one adds a new bin in Step 3 of Bin-PTAS, then it means that all bins have sizes larger than 1. Thus, the sum of the capacities (sizes) of these bins is larger than its number and so any optimal solution must contain one bin more. □

PTAS for MS（1）

- BIN-P和MS是如何相互转化的？

$$Opt_{\text{Bin-P}}\left(\frac{p_1}{d}, \frac{p_2}{d}, \ldots, \frac{p_n}{d}\right) \leq m \Leftrightarrow Opt_{\text{MS}}(I, m) \leq d.$$



- 解MS的基本思路？

PTAS for MS（2）

Algorithm 4.3.6.7.

Input: $((I, m), \varepsilon)$, where $I = (p_1, \ldots, p_n)$, for some $n \in \mathbb{N}$, $p_1, \ldots, p_n$, m are positive integers, and $\varepsilon > 0$.

Step 1: Compute $\underline{ATLEAST := \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i, \max\{p_1, \ldots, p_n\}\right\}}$;
Set $LOWER := ATLEAST$;
$UPPER := \underline{2 \cdot ATLEAST}$; → example 4.2.1.2 (书上4.3.4.14印错了）
$k := \lceil \log_2(4/\varepsilon) \rceil$.

Step 2: **for** $i = 1$ **to** k **do**
  do begin $d := \frac{1}{2}(UPPER + LOWER)$;
    call Bin-PTAS$_{\varepsilon/2}$ on the input $\left(\frac{p_1}{d}, \frac{p_2}{d}, \ldots, \frac{p_n}{d}\right)$;
    $c := cost\left(\text{Bin-PTAS}_{\varepsilon/2}\left(\frac{p_1}{d}, \ldots, \frac{p_n}{d}\right)\right)$
    if $c > m$ **then** $LOWER := d$
      else $UPPER := d$
  end

此算法代价最大的部分，执行常量次Bin-PTAS。

Step 3: Set $\underline{d^* := UPPER}$;
call Bin-PTAS$_{\varepsilon/2}$ on the input $\left(\frac{p_1}{d^*}, \ldots, \frac{p_n}{d^*}\right)$.

Output: Bin-PTAS$_{\varepsilon/2}\left(\frac{p_1}{d^*}, \ldots, \frac{p_n}{d^*}\right)$.

本节的来源

- Dorit S. Hochbaum, David B. Shmoys. Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results. *Journal of the ACM*, 34(1):144—162, 1987.

Bin-Packing 问题

你能想到的解决BIN-P问题的方法？

Greedy is good

Bin-Packing 问题

- Next Fit (Decreasing)
  - If the item fits in the same bin as the previous item, put it there. Otherwise, open a new bin and put it in there.
- First Fit (Decreasing) **FFD** $\boxed{\frac{11}{9} + \frac{4}{m}}$
  - Put each item as you come to it into the oldest (earliest opened) bin into which it fits.
- Worst Fit (Decreasing)
  - Put each item into the emptiest bin among those with something in them. Only start a new bin if the item doesn't fit into any bin that's already been started.

Bin-Packing 问题

- No approximation algorithm having a guarantee of 3/2.
  - Reduction from the set partition, an NP-complete problem.
- Set Partition
  - Whether a given multiset S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2.

BIN-P和MS

- 一维装箱问题：将(0,1]容量的物品放到单位大小的bin中，优化(最小化)bin数目

  - **[BIN-P]** Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24, 2017.
  - **[MS]** Gerhard J Woeginger. The open shop scheduling problem. *LIPIcs-Leibniz International Proceedings in Informatics*, volume 96. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

- MS → $PDm||Obj$
  - MS: $PDm||C_{max}$
  - $PDm||\sum w_j C_j$

近似算法小结

- 近似算法的分类
  - FPTAS：(S)KP
  - PTAS：*Euc*-TSP、MS
  - 常数近似：MIN-VCP、MAX-SAT、Δ-TSP
  - log函数近似：SCP
  - 多项式函数近似：TSP、MAX-CL

- 近似算法的稳定性
- 对偶近似算法

求解难问题的方法

- 你还记得这些方法吗？它们的要点分别是什么？
  - 伪多项式
  - 分支限界
  - 局部搜索
  - 松弛算法
  - 贪心策略