(8) Multiple Machine Models.

Notation:

Recall we write $\alpha|\beta|\gamma$, where α = mlc environment

Common choices for α we will use in defining multiple machine environments are:

- P: multiple identical mlc's OR parallel mlc's
  Each job j takes time p_j to be processed on any mlc
- R: unrelated mlc's:
  job j requires time p_{ij} to be processed on machine i.
  (P is the special case where $p_{ij} = p_j \ \forall i, j$)
- Q: related mlc's
  Each mlc i has speed $s_i \geq 0$ and so the processing time for job j on mlc i is $p_{ij} = p_i / s_i$.
  (P is the special case where $s_i = 1 \ \forall i$)

In each of these cases, a job j only has to be processed on 1 mlc.

Sometimes, we write P_m, Q_m, or R_m, where m denotes the number of mlc's.

↪ P||pmtn|Cmax

P|pmtn|C_{max}:

We define:

$C_{max} := \max_j C_j$

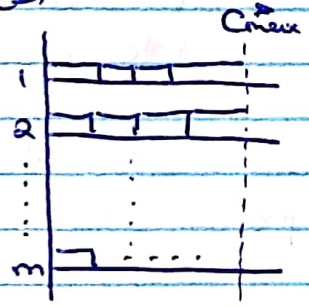
often called the makespan of a schedule.

Note: Preemption in a multiple m/c environment means that job j can be interrupted and resumed later on any m/c. However, at any point in time, at most 1 m/c can be processing the job.

Let us first find a lower bound on C_{max}:

(1) $C^*_{max} \geq p_j \ \forall j \equiv C^*_{max} \geq p_{max} := \max_j p_j$.

(2)



In the time interval $[0, C^*_{max}]$, we have mC^*_{max} total processing capacity on the m m/c's, and since there are no jobs past that point!

$$mC^*_{max} \geq \sum_{j=1}^{n} p_j$$

So, (1) and (2) give:

$$C^*_{max} \geq LB := \max\left\{ \sum_{j=1}^{n} p_j / m, \ p_{max} \right\}$$

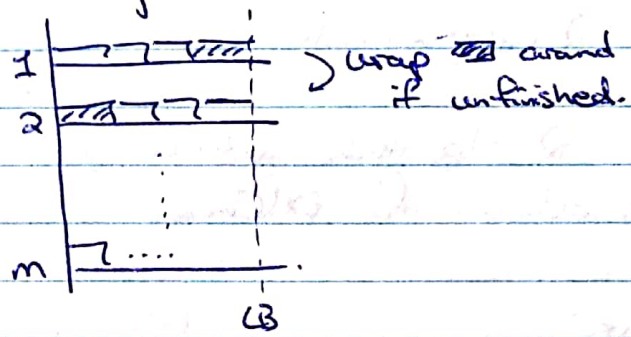
load-based lower bound ↑ job-based lower bound ↑

Con't

P|pmtn|C_{max} : (cont'd)

McNaughton's Wraparound Rule:

Consider jobs in arbitrary order and begin packing them in the interval $[0, LB]$ on the m/c's. When we reach LB on m/c i, we move to m/c i+1 and resume with the unfinished portion of the current job.



Claim 8.1: We finish processing all jobs using this rule

We have enough space to pack all jobs since $m \cdot LB \geq \sum P_i$

Claim 8.2: No job is processed on multiple m/c's at the same point of time

$LB \geq P_{max}$, so if we have multiple chunks of a job, they cannot overlap.

Using this rule, we have an algorithm that obtains an optimal schedule for P|pmtn|C_{max}.

$\Rightarrow P||C_{max}$

P||Cmax:

We can obtain an easy non-preemptive algorithm by simply not preempting a job if it wraps around (in the schedule for P|pmtn|Cmax). Since jobs have a max processing time of p_{max}, this gives:

$$LB + p_{max} \leq 2LB$$

which is a 2-approximation.

We will show 2 algorithms which give even better approximations for P||Cmax.

List Scheduling:

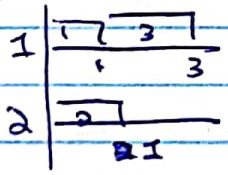
Consider a list of jobs in any order. Schedule the first unscheduled job on the list on the first available mc.

Ex:

$m = 2$, $P_1, P_2 = 1$, $P_3 = 2$

List = [1, 2, 3]

Using list scheduling this gives:



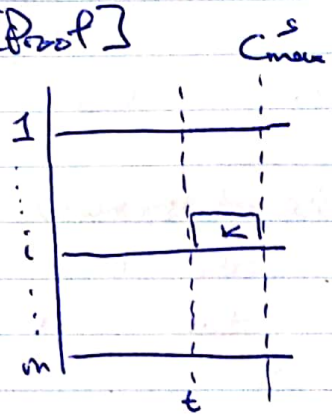
→ Notice that the makespan is 3 (But, clearly, the OPT makespan is 2).

Theorem 8.3:

List Scheduling is a $(2 - \frac{1}{m})$-approximation algorithm for P||Cmax

↳ [Proof]

[Proof]



Let S be the schedule constructed
Let k be the job to finish last in S, i.e. $C^S_k = C^S_{max}$
Let t be the time where k begins processing

In $[0,t]$, all m/c's are busy processing jobs other than k (Otherwise, we would schedule k earlier).
So,

$m \cdot t \leq$ Total processing time of jobs available other than k.

i.e.

$$m \cdot t \leq \sum_{j \neq k} P_j = \sum_{j=1}^{n} P_j - P_k$$

Then,

$$C^S_{max} = t + P_k \leq \underbrace{\sum_{j=1}^{n} P_j / m}_{\leq LB} - P_k/m + P_k$$

$$\leq LB - \underbrace{P_k}_{\leq P_{max} = LB}\left(1 - \frac{1}{m}\right)$$

$$\leq 2LB$$

So, we have a 2-approximation.

□.

↳ Another algorithm

P||Cmax (cont)

List Scheduling w/ Longest Processing Time (LPT) Ordering:

(We abbreviate this to LPT-Schedule(Rule))

Sort jobs in decreasing P_j order and run LIST-SCHEDULING (LS) with this order.

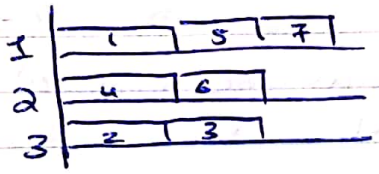
Ex:

$m=3$

j	1	2	3	4	5	6	7
P_j	5	4	4	5	3	3	3

→ This gives the ordering: 1, 4, 2, 3, 5, 6, 7.

S:



Theorem 8.4:

LS with the LPT-rule is a $(\frac{4}{3}-\frac{1}{3m})$-approx. for P||Cmax

[Proof]

Let S: Schedule constructed,

Let k: the last job that completes in S,

so $C_k^S = C_{max}^S$

Let t: the point of time where k begins processing.

Let $J' = \{j : P_j \geq P_k\}$

→ We can focus on J' since under the LPT rule, the last job to start processing is the shortest job. If it also finishes last, then [redacted] = we have that job k is the last job. Otherwise, we can delete this job without affecting the makespan

Cont

[Proof] (cont'd)

Since it suffices to only focus on J', we have:

- The LPT-schedule for J' is the LPT-schedule for J up to k, call this S'.
- $C_{max}^{S'} = C_{max}^{S}$
- We can show that:

$$C_{max}^{S'} \leq \left(\frac{4}{3} - \frac{1}{3m}\right) OPT(J')$$

since $OPT(J') \leq OPT(J)$

Further, because of the simplification, we can say:

- $P_k = \min_{j \in J'} P_j$, and
- k is the last job to start and finish in S'.

Write $OPT' = OPT(J')$ and consider 2 cases:

Case 1: $P_k \leq OPT'/3$

From proof of Theorem 8.3, we have:

$$C_{max}^{S'} \leq \sum_{j \in J'} P_j + P_k\left(1 - \frac{1}{m}\right)$$

$$\leq OPT' + \frac{OPT'}{3}\left(1 - \frac{1}{m}\right) = \left(\frac{4}{3} - \frac{1}{3m}\right) OPT'$$

Case 2: $P_k > OPT'/3$.

So, $P_j > OPT'/3 \ \forall j$ since $P_j \geq P_k \ \forall j$.

$\Rightarrow$ Every optimal schedule (for J') schedules ≤ 2 jobs per mlc. So $n' := |J'| \leq 2m$.

(a) $n' \leq m$: LPT will schedule at most one job per machine and such a schedule must be optimal.

Con't

[Proof] (cont'd)

(b) $m < n' \leq 2m$:

First, we claim that there is an optimal schedule that schedules at least one job per mlc. This is true since we can always move a job from an mlc to another, having 0 jobs, without increasing the makespan. Then, since $n' \geq m$, we have at least one job on each mlc.

So, there is an optimal schedule where there are [redacted] 1 or 2 jobs per mlc. We will show that we can define a "nice structure" for these schedules and argue that this must be an LPT schedule.

Label the jobs, in the optimal schedule, on mlc i as follows:

$(i,1)$, $(i,2)$, where $P_{(i,1)} \geq P_{(i,2)}$ with the convention where if i has only one job, then $(i,2) = \text{NULL}$ and $P_{(i,2)} = 0$.

Reorder the machines so that:

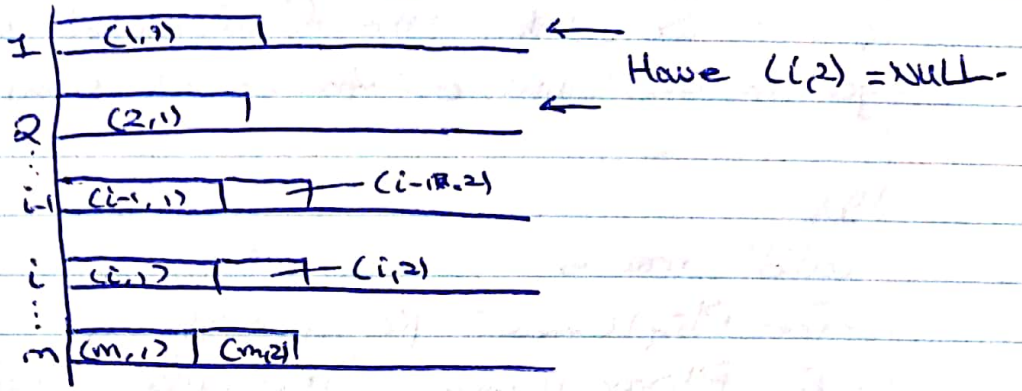
$P_{(i,1)} \geq P_{(i,2)} \geq \ldots \geq P_{(i,m)}$

Further, we can also assume that if $i^* < i'$, then $P_{(i',2)} \geq P_{(i,2)}$, with this ordering. Since, we can always interchange the 2 without affecting the makespan.

Cont'd

Proof 3 (cont'd)

So, we have the following structure:



So, $P_{(1,1)} \geq P_{(2,1)} \geq \cdots \geq P_{(m,1)} \geq P_{(m,2)} \geq \cdots$
$\cdots \geq P_{(i,2)} \geq \cdots \geq P_{(i,2)}$

We claim all LPT schedules will look like this (and hence are also optimal). To show this, LPT schedules cannot assign a 3rd job to any m/c.

Suppose i' is the m/c on which LPT assigns a 3rd job for the first time.

The rest of this argument makes no sense, and is unnecessary.
Try a proof using contradiction.
i.e. LPT produces a $> (\frac{4}{3} - \frac{1}{3m})$-approx.

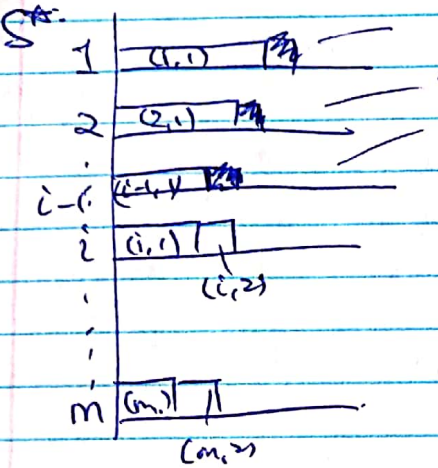
Hilroy

Theorem 8.4

The offending proof

Proof (cont'd)

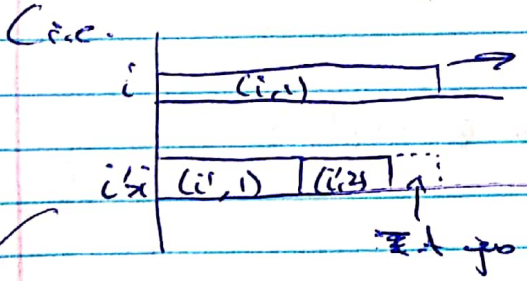
So, now we have the following pic structure of an optimal schedule for J



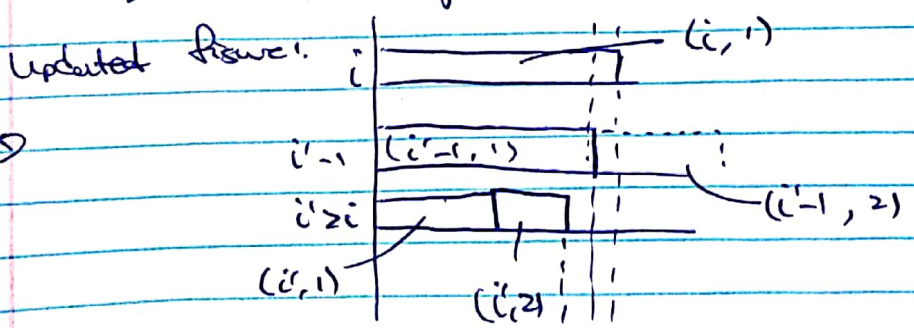
So,

$$P_{(1,1)} \geq P_{(2,1)} \geq \cdots P_{(i-1,1)} \geq P_{(i,1)} \geq \cdots$$
$$\cdots \geq P_{(m,1)} \geq P_{(m,2)} \geq P_{(m-1,2)} \geq \cdots \geq P_{(i,2)}$$

We want to show that S^* is an LPT-schedule. Back for this, we have to show that LPT does not assign a 3rd job on any machine.



Suppose that i' is the m/c on which LPT assigns a 3rd job for the first time



Hilroy

Claim: $(i'-1, 2)$ is not null.

If $(i'-1, 2)$ is not null, then # of jobs n', is $\leq$ [looking at OPT schedule] $1 \cdot (i'-1)$ [M/C's $1, \ldots i'-1$ have 1 job] $+ 2(m-i'+1)$ [m/c's $i', \ldots, m$ have ≤ 2 jobs]

But looking at OPT schedule $n' > 1 - i' + 2(m-i'+1)$ since it has assigned ≥ 1 jobs on $1, \ldots, i'-1$, and 2 jobs on $i', \ldots, m$, and is still left with more jobs

Given our claim, the load on $i'-1$ in S^* is $P_{(i'-1,1)} + P_{(i'-1,2)}$

$\geq P_{(i',1)} + P_{(i',2)} > \frac{OPT'}{3}$

Since OPT is assigning a 3rd job on i'

$\Rightarrow > 2 \cdot \frac{OPT'}{3}$

$\Rightarrow P_{(i'-1,1)} + P_{(i'-1,2)} > \frac{2}{3} OPT'$,

which is a contradiction.

Generalized List Scheduling:

Consider a list of jobs in any given order. Schedule the first valid unscheduled job in the list on the first available mc

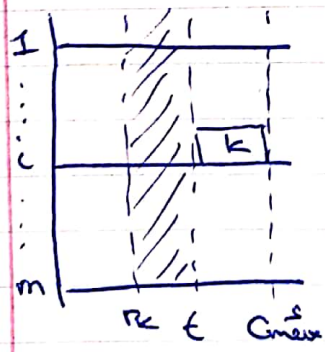
Note:
"Valid" means:
- For $P|r_j|C_{max}$: job is released
- For $P|prec|C_{max}$: Predecessors have completed
- For $P|r_j, prec|C_{max}$: Released and all predecessors have completed

Theorem 8.5:
(Generalized) LS is a 2-approx. for $P|r_j|C_{max}$.

[Proof]
We first refine our lower bound to account for release dates:

$$C^*_{max} \geq \max\left\{\sum p_j/m,\ \max_j (r_j + p_j)\right\}.$$

Call this LB'.



Let S: Schedule constructed,
Let k: job to complete last in S
Let t: time where k begins processing

Cont'd

[Proof] (cont'd)

In the interval $[r_k, t]$, mc's $1, \ldots, m$ must be busy processing jobs other than k, since if not we could have scheduled k earlier. In other words:

$$m(t - r_k) \leq \sum_{j \neq k} p_j \leq \sum_{j=1}^{n} p_j$$

So,

$$\begin{aligned} C_{max}^{S} &= t + p_k \\ &= (r_k + p_k) + (t - r_k) \\ &\leq \max_j (r_j + p_j) + \sum_{j=1}^{n} p_j / m \\ &= 2 \cdot LB' \end{aligned}$$

□.

Theorem 8.6:

Generalized LS is a 2-approx. for $P|prec|C_{max}$

[Proof]

Again, we refine our lower bound to account for precedence constraints.

Suppose we have a chain of jobs: $j_1 \to j_2 \to \ldots \to j_n$ then,

$$C_{max}^{*} \geq p_{j_1} + p_{j_2} + \ldots + p_{j_n}.$$

So,

$$C_{max}^{*} \geq \max \left\{ \sum_{j=1}^{n} p_j / m, \max_{\text{all chains } j_1 \to \ldots \to j_r} (p_{j_1} + \ldots + p_{j_r}) \right\}$$

← Call this "LB"

Let S and k be as defined in Theorem 8.5, and let t_k be the start time of k.

Cont'd

[Proof] (Cont'd)

We will construct a chain $C: k_r \to \cdots \to k_1 := k$, s.t. at every time $t \in [0, b_3]$, either:

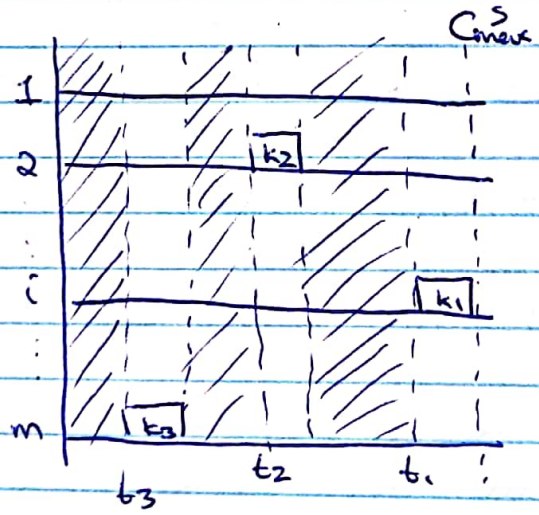
(1) Some job of C is processed at time t; or

(2) All m/c's are busy.

Let k_2 be the job s.t. $k_2 \to k$ and k_2 completes last among all predecessors of k_2. Let t_2 be the start time of k_2. (If k_1 has no predecessors, set $k_2 = \text{NULL}$, $t_2 = 0$ and $P_{k_2} = 0$)

And continue this way: In general, k_{i+1} is the predecessor of k_i that completes last among all predecessors of k_i. t_{i+1} is the start time of k_{i+1}, and if k_i has no predecessors, set $k_{i+1} = \text{NULL}$, $t_{i+1} = 0$, $P_{k_{i+1}} = 0$

i.e.



This gives a chain: $k_r \to k_{r-1} \to \cdots \to k_1$ with $k_{r+1} = \text{NULL}$, $t_{r+1} = 0$, $P_{k_{r+1}} = 0$.

$\Leftarrow$ Cont

[Proof] (cont'd)

For any $d = 1, \ldots, r$, all mc's must be busy within $[t_{k_{d+1}} + P_{k_{d+1}}, t_2]$, since all predecessors of k_d will have been completed, so if the mc's are not all busy we can schedule k_d at an earlier time. And so:

$$m \cdot \left(\begin{array}{c} \text{total length of all} \\ [t_{k_{d+1}} + P_{k_{d+1}}, t_2] \\ \text{intervals} \end{array} \right) \leq \sum_{\substack{k \text{ not} \\ \text{in chain } C}} P_k \leq \sum_{i=1}^{n} P_i$$

Then,

$$C^S_{max} - (\text{total length of all } [t_{k_{d+1}} + P_{k_{d+1}}, t_2] \text{ intervals}) + (P_{k_1} + \cdots + P_{k_r})$$

$$\leq \sum_{i=1}^{n} P_i / m + \max_{\text{chain}} P_{i_1} + \cdots + P_{i_n}$$

$$= 2 \cdot LB''.$$

□.

$\Rightarrow Q \| C_{max}$

Q||C_{max}:

Recall: Each m/c i has speed $s_i > 0$, and so takes time p_i/s_i to complete job j on m/c i.

Again, we begin by refining LB to account for the speeds s_i.

(1) In time $[0, C^*_{max}]$, each machine i can do $s_i(C^*_{max})$ amount of work.
So, in total, all m machines can do: $\left(\sum_{i=1}^{m} s_i\right) C^*_{max}$ amount of work. And this had better be $\geq \sum_{i=1}^{n} p_i$.
So, we have:

$$\left(\sum_{i=1}^{m} s_i\right) C^*_{max} \geq \sum_{i=1}^{n} p_i$$

$$\Rightarrow C^*_{max} \geq \sum_{i=1}^{n} p_i \Big/ \sum_{i=1}^{m} s_i.$$

(2) Order m/c's so that $s_1 \geq s_2 \geq \dots \geq s_m$
Order jobs so that $p_1 \geq p_2 \geq \dots \geq p_n$.
The fastest m/c should be able to complete the longest job by C^*_{max}:
So:

$$s_1 \cdot C^*_{max} \geq p_1 \Rightarrow C^*_{max} \geq p_1/s_1$$

And, in general, for any $k \leq m$, the k fastest m/c's should be able to complete the k longest jobs by C^*_{max}
So, we have:

$$C^*_{max} \geq \max_{k=1,\dots,m} \frac{\sum_{i=1}^{k} p_i}{\sum_{i=1}^{k} s_i}.$$

Q||C_{max}: (cont'd)

Combining (1) and (2), we have:

$$C^*_{max} \geq \max\left\{ \sum_{i=1}^{n} P_i / \sum_{i=1}^{m} S_i \;,\; \max_{k \leq m} \sum_{i=1}^{k} P_i / \sum_{i=1}^{k} S_i \right\}.$$

LS w/ LPT ordering (for Q||C_{max}):

Order jobs in ↓ P_j order (so $P_1 \geq P_2 \geq \cdots \geq P_n$)

Repeatedly schedule the first unscheduled job on the mc on which it finishes earliest (given currently scheduled jobs).

Theorem 8.7:

LS with LPT rule is a $(2-\frac{1}{m})$-approx. for Q||C_{max}

[Proof]

Let S: Schedule constructed, k: job that finishes last. And, as with P||C_{max}, we can assume that $P_k = \min_{j=1,\ldots,n} P_j = P_n$.

Case 1: $m \leq n$

Since n was scheduled on mc i, we know that ∀mc i', completion time if job n was scheduled on mc i' $\geq C^S_{max}$:

i.e.

$$\forall i' = 1, \ldots, m \quad \left(P_n + \sum_{\substack{j \neq i' \\ j \neq n}} P_j \right) / S_{i'} \geq C^S_{max}$$

So summing all m equations, we get:

$$mP_n + \sum_{j \neq n} P_j \geq C^S_{max} \cdot \sum_{i=1}^{m} S_i$$

$$\Rightarrow C^S_{max} \cdot \sum_{i=1}^{m} S_i \leq \sum_{j=1}^{n} P_j + (m-1)P_n = \sum_{j=1}^{n} P_j + \left(1-\frac{1}{m}\right) m P_n.$$

$$\leq \sum_{j=1}^{n} P_j + \left(1-\frac{1}{m}\right) n P_n.$$

→ Cont'd

Hilroy

Proof (cont'd)

$$C^S_{max} \sum_{i=1}^{m} s_i \le \sum_{i=1}^{n} p_i + \left(1-\frac{1}{m}\right) n p_n$$

$$\le \sum_{i=1}^{n} p_i + \left(1-\frac{1}{m}\right) \sum_{i=1}^{n} p_i \quad (p_n \le p_i)$$

$$\Rightarrow C^S_{max} \le \frac{\sum_{i=1}^{n} p_i}{\sum_{i=1}^{m} s_i} \left(2-\frac{1}{m}\right) \le \left(2-\frac{1}{m}\right) C^*_{max}$$

Case 2: $m > n$

So, $m-n$ m/c's will be idle, in fact, the machines from $n+1, \ldots, m$ will be idle.

Using the equation from Case 1, and adding $\forall i = 1, \ldots, n$.

$$\left(\sum_{i=1}^{n} s_i\right) \cdot C^S_{max} \le \sum_{i=1}^{n} p_i + (n-1) p_n$$

$$\le \sum_{i=1}^{n} p_i \cdot \left(2-\frac{1}{n}\right)$$

$$\Rightarrow C^S_{max} \le \left(2-\frac{1}{n}\right) \left(\sum_{i=1}^{n} p_i \Big/ \sum_{i=1}^{n} s_i\right)$$

$$\le \left(2-\frac{1}{m}\right) C^*_{max} \quad (\text{Since } n < m)$$

□.