Improved Semi-Online Makespan Scheduling with a Reordering Buffer

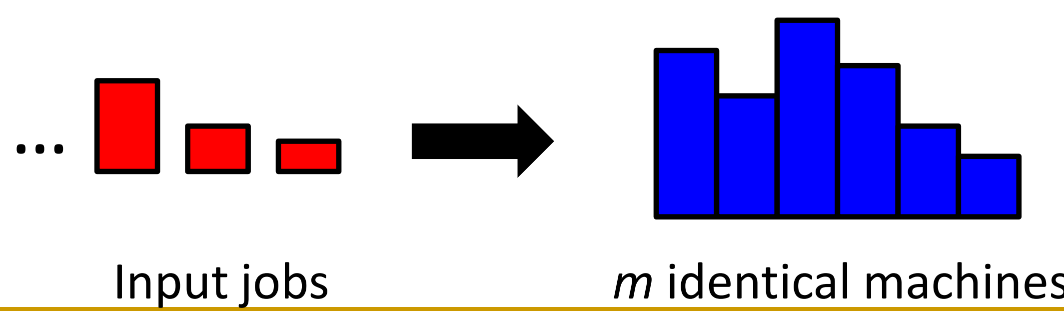
Hongyang Sun (Speaker)

Joint work with *Rui Fan*

Nanyang Technological University, Singapore

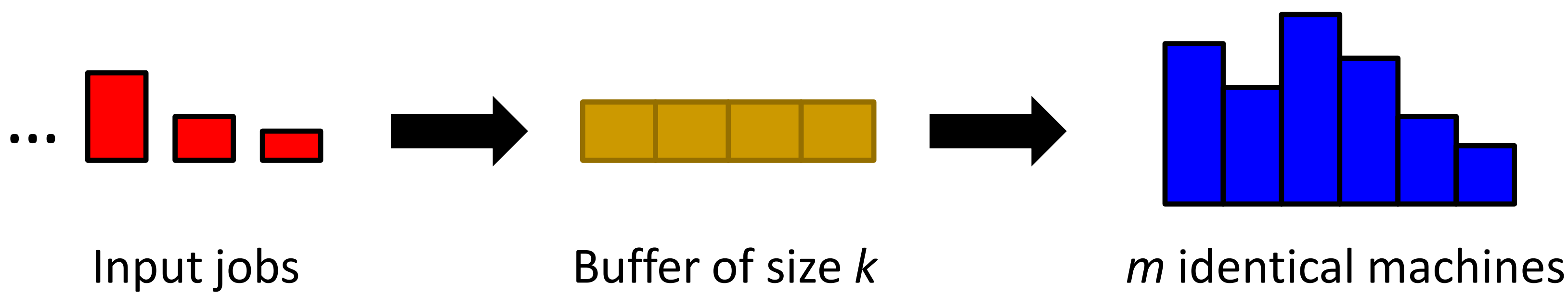
Background

- **Classical online scheduling**
  - Schedule a sequence of jobs arriving one by one on m identical machines to minimize makespan
  - List scheduling algorithm (Graham 1966)
    - Assign arriving job on a machine with least load
    - $(2-1/m)$-competitive
  - Best known competitive ratio of deterministic algorithm [1.88, 1.9201] for large m



Background

- **Semi-online scheduling with reordering buffer**
  - A buffer of limited size *k* (independent of input size) can be used to store and reorder jobs
    - Store: If buffer is not full, we can admit a new job into the buffer without assigning any job to any machine
    - Reorder: If buffer is full, we can select any job from the buffer or the arriving job and assign it to a machine



Results

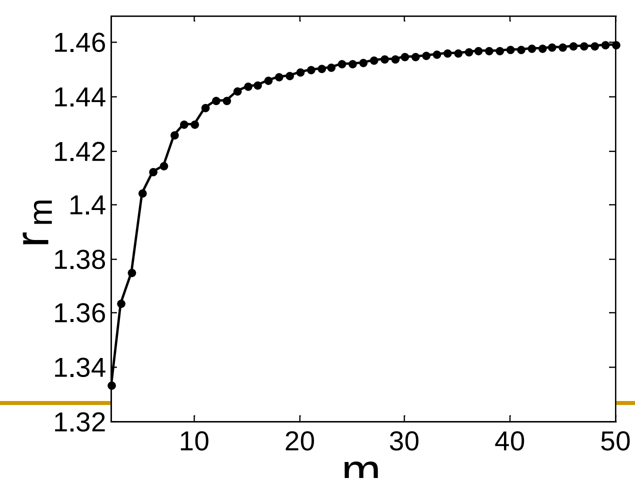
- **Semi-online scheduling with reordering buffer**
  - For $m = 2$, optimal comp. ratio = $4/3$ (Kellerer et al. 1997 & Zhang 1997) using $k = 1$, the smallest possible buffer size

Results

- **Semi-online scheduling with reordering buffer**
  - For $m = 2$, optimal comp. ratio = $4/3$ (Kellerer et al. 1997 & Zhang 1997) using $k = 1$, the smallest possible buffer size
  - For general m, optimal comp. ratio (lower bound) = r_m (Englert, Ozmen and Westermann 2008)
    - r_m is the solution to the following equation
      $\lceil (r_m - 1)m/r_m \rceil \cdot r_m/m + (r_m - 1) \cdot \sum_{i=\lceil (r_m-1)m/r_m \rceil}^{m-1} 1/i = 1$

Results

- **Semi-online scheduling with reordering buffer**
  - For $m = 2$, optimal comp. ratio = $4/3$ (Kellerer et al. 1997 & Zhang 1997) using $k = 1$, the smallest possible buffer size
  - For general m, optimal comp. ratio (lower bound) = r_m (Englert, Ozmen and Westermann 2008)
    - r_m is the solution to the following equation
      $$\lceil (r_m - 1)m/r_m \rceil \cdot r_m/m + (r_m - 1) \cdot \sum_{i=\lceil (r_m-1)m/r_m \rceil}^{m-1} 1/i = 1$$
    - r_m is a monotonically increasing function of m
    - $r_2 = 4/3$, and $\lim_{m\to\infty} r_m \approx 1.4659$



Results

- **Semi-online scheduling with reordering buffer**
  - For general m, optimal comp. ratio (lower bound) = r_m (Englert, Ozmen and Westermann 2008)
    - $\boldsymbol{k = \Theta(m)}$ is necessary and sufficient to achieve r_m
    - A r_m-comp. algorithm was proposed with a reordering buffer of size $\boldsymbol{k = (1+2/r_m)m \approx 2.364m}$ for large m
    - A lower bound of $\boldsymbol{k = m/2}$ on the buffer size

Results

- **Semi-online scheduling with reordering buffer**
  - For general m, optimal comp. ratio (lower bound) = r_m (Englert, Ozmen and Westermann 2008)
    - $\boldsymbol{k = \Theta(m)}$ is necessary and sufficient to achieve r_m
    - A r_m-comp. algorithm was proposed with a reordering buffer of size $\boldsymbol{k = (1+2/r_m)m \approx 2.364m}$ for large m
    - A lower bound of $\boldsymbol{k = m/2}$ on the buffer size
  - What is the exact buffer size required to achieve r_m?

Results

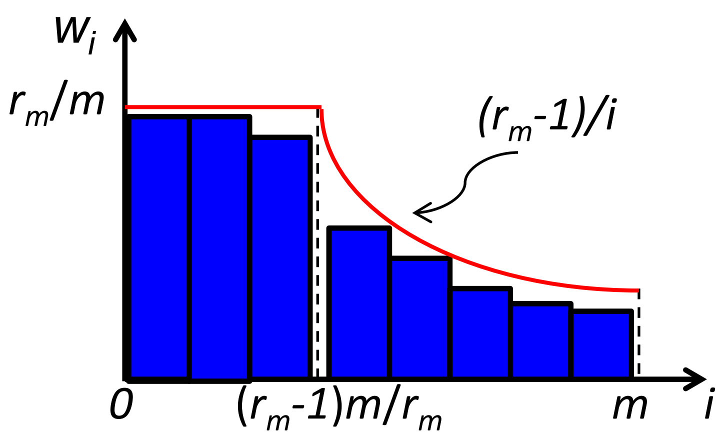
- **Semi-online scheduling with reordering buffer**
  - For general m, optimal comp. ratio (lower bound) = r_m (Englert, Ozmen and Westermann 2008)
    - $\boldsymbol{k = \Theta(m)}$ is necessary and sufficient to achieve r_m
    - A r_m-comp. algorithm was proposed with a reordering buffer of size $\boldsymbol{k = (1+2/r_m)m \approx 2.364m}$ for large m
    - A lower bound of $\boldsymbol{k = m/2}$ on the buffer size
  - What is the exact buffer size required to achieve r_m?
  - **Our result improves the required buffer size**
    - A r_m-comp. algo. with a buffer size $\boldsymbol{k = (5-2r_m)m \approx 2.068m}$ for large m, improving the previous result by $\boldsymbol{\approx 0.296m}$

Outline

- Lower bound $\boldsymbol{r_m}$
- A scheduling framework to get optimal comp. ratio
  - Buffer size $\boldsymbol{k = (1+2/r_m)m \approx 2.364m}$ (Englert et al.)
  - Buffer size $\boldsymbol{k = (5-2r_m)m \approx 2.068m}$ (Our result)
- Tradeoff between comp. ratio and buffer size
- Remarks

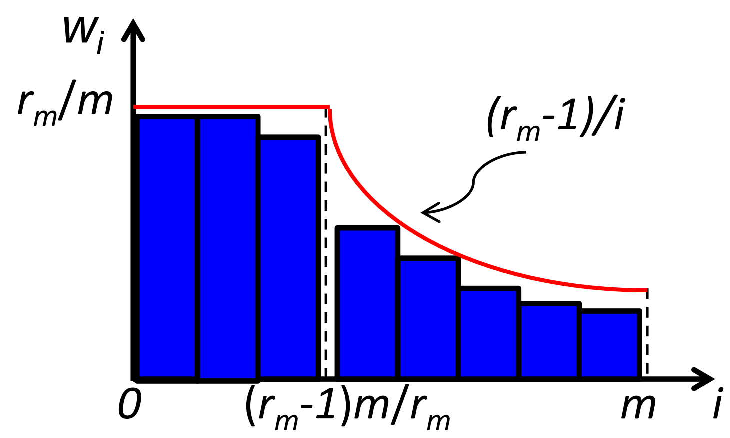
Lower bound r_m

- Consider the following load profile (weight w_i)
  - $\boldsymbol{w_i = \min\{r_m/m, (r_m-1)/i\}}$, where $\boldsymbol{\sum w_i = 1}$



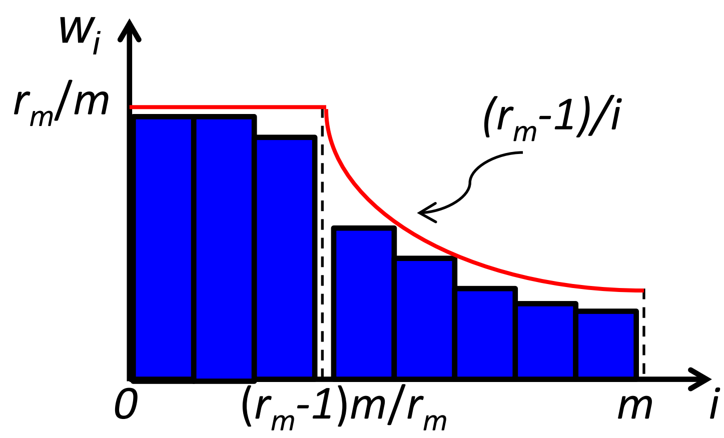
Lower bound r_m

- Consider the following load profile (weight w_i)
  - $\boldsymbol{w_i = \min\{r_m/m, (r_m-1)/i\}}$, where $\boldsymbol{\sum w_i = 1}$
- Arbitrarily small jobs of total size $1+\varepsilon$ arrive
  - Buffer contains size = ε, and total assigned job size = 1



Lower bound r_m

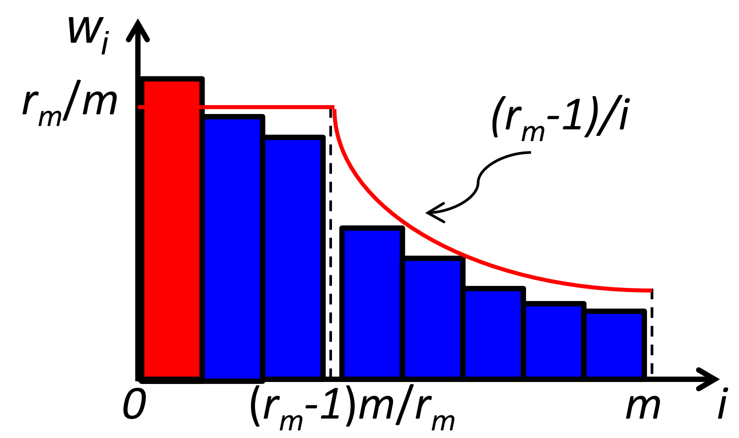
- Consider the following load profile (weight w_i)
  - $\boldsymbol{w_i = \min\{r_m/m, (r_m-1)/i\}}$, where $\boldsymbol{\sum w_i = 1}$
- Arbitrarily small jobs of total size $1+\varepsilon$ arrive
  - Buffer contains size = ε, and total assigned job size = 1
- There must be a machine j with load $\geq w_j$,otherwise $\sum w_i < 1$



Lower bound r_m

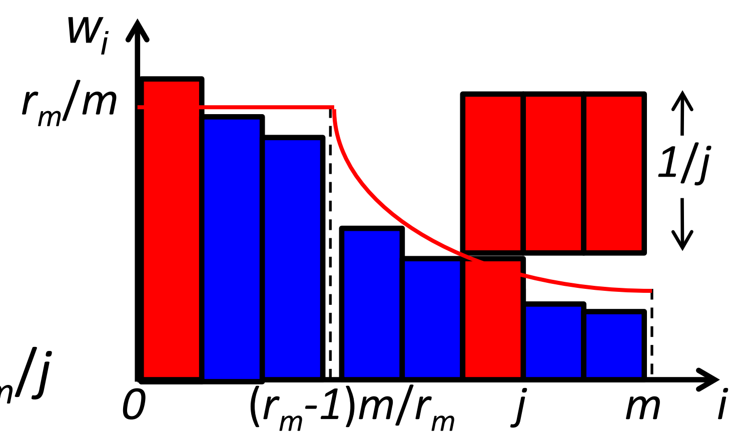
- Consider the following load profile (weight w_i)
  - $\boldsymbol{w_i = \min\{r_m/m, (r_m-1)/i\}}$, where $\boldsymbol{\sum w_i = 1}$
- Arbitrarily small jobs of total size $1+\varepsilon$ arrive
  - Buffer contains size = ε, and total assigned job size = 1
- There must be a machine j with load $\geq w_j$,otherwise $\sum w_i < 1$
  - If $w_j = r_m/m$, no more job arrives

    → OPT = $1/m$, ALG $\geq r_m/m$



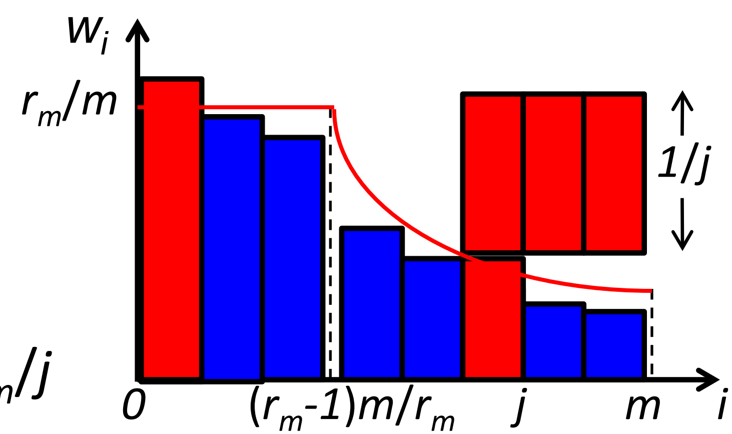
Lower bound r_m

- Consider the following load profile (weight w_i)
  - $\mathbf{w_i = \min\{r_m/m, (r_m-1)/i\}}$, where $\sum w_i = 1$
- Arbitrarily small jobs of total size $1+\varepsilon$ arrive
  - Buffer contains size = ε, and total assigned job size = 1
- There must be a machine j with load $\geq w_j$,otherwise $\sum w_i < 1$
  - If $w_j = r_m/m$, no more job arrives
    → OPT = $1/m$, ALG $\geq r_m/m$
  - If $w_j = (r_m-1)/j$, m-j large jobs of size $1/j$ arrive
    → OPT = $1/j$, ALG $\geq (r_m-1)/j + 1/j = r_m/j$



Lower bound r_m

- Consider the following load profile (weight w_i)
  - $\boldsymbol{w_i = \min\{r_m/m, (r_m-1)/i\}}$, where $\sum w_i = 1$
- Arbitrarily small jobs of total size $1+\varepsilon$ arrive
  - Buffer contains size = ε, and total assigned job size = 1
- There must be a machine j with load $\geq w_j$,otherwise $\sum w_i < 1$
  - If $w_j = r_m/m$, no more job arrives
    → OPT = $1/m$, ALG $\geq r_m/m$
  - If $w_j = (r_m-1)/j$, $m-j$ large jobs of size $1/j$ arrive
    → OPT = $1/j$, ALG $\geq (r_m-1)/j + 1/j = r_m/j$
- In both cases, competitive ratio $\geq r_m$



A scheduling framework to get optimal comp. ratio

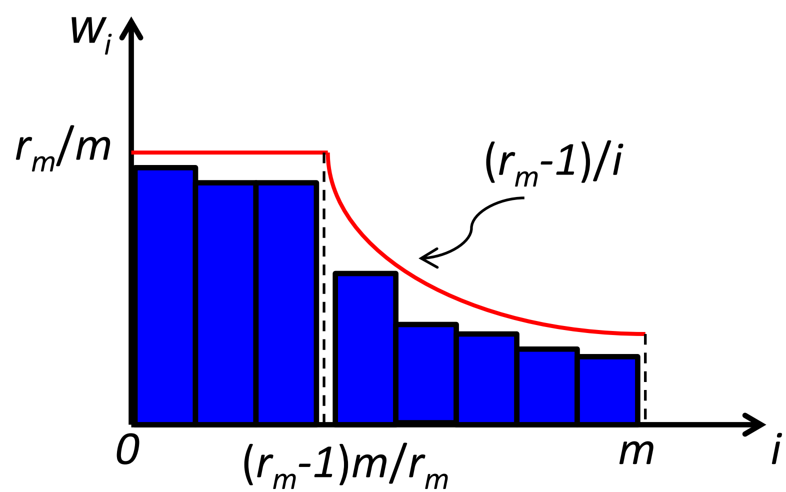
- **Three phases:**
  - **(1) Initial phase:** Admit *k* jobs into buffer w/o assignment

A scheduling framework to get optimal comp. ratio

- **Three phases:**
  - **(1) Initial phase:** Admit k jobs into buffer w/o assignment
  - **(2) Iterative phase**: Admit a new job, and pick a smallest job and assign it to some machine j

A scheduling framework to get optimal comp. ratio

- **Three phases:**
  - **(1) Initial phase:** Admit *k* jobs into buffer w/o assignment
  - **(2) Iterative phase**: Admit a new job, and pick a smallest job and assign it to some machine *j*
    - Choice of the machine depends on the algorithm
    - Maintain a profile of machine loads related to $\boldsymbol{w_i = \min\{r_m/m, (r_m\text{-}1)/i\}}$, where normalized total area = 1, i.e., $\boldsymbol{\sum w_i = 1}$

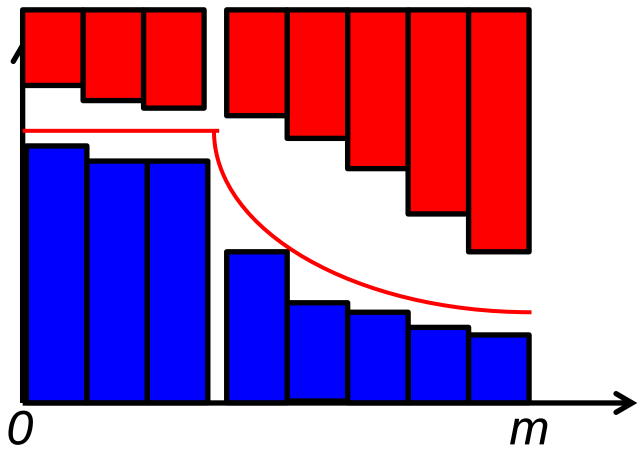


A scheduling framework to get optimal comp. ratio

- **(3) Final phase**

A scheduling framework to get optimal comp. ratio

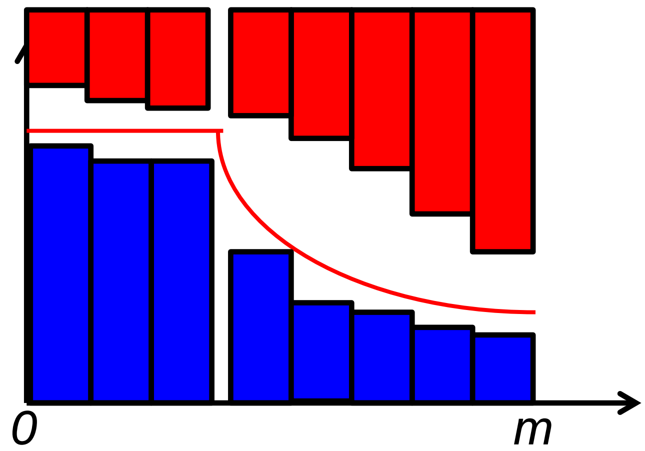
- **(3) Final phase**
  - **1st Step**: Large jobs (size > *1/3*•OPT)
    - Make an optimal schedule (LPT)
    - Sort in ascending order of size
    - Place them on current schedule



A scheduling framework to get optimal comp. ratio

- **(3) Final phase**
  - **1st Step**: Large jobs (size > *1/3*•OPT)
    - Make an optimal schedule (LPT)
    - Sort in ascending order of size
    - Place them on current schedule

    → Mathematics can ensure the optimal comp. ratio r_m

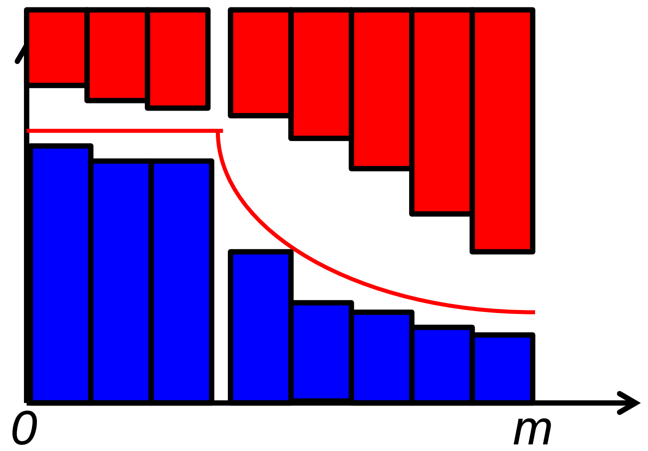


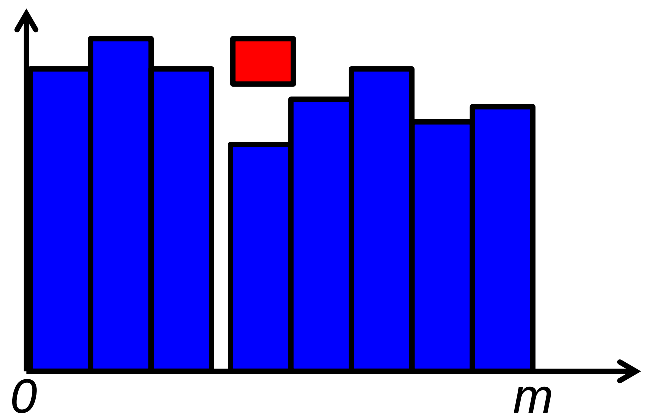
A scheduling framework to get optimal comp. ratio

- **(3) Final phase**
  - **1st Step**: Large jobs (size > $1/3 \cdot$OPT)
    - Make an optimal schedule (LPT)
    - Sort in ascending order of size
    - Place them on current schedule

    → Mathematics can ensure the optimal comp. ratio r_m
  - **2nd Step**: Small jobs (size ≤ $1/3 \cdot$OPT)
    - Place one by one greedily
    - ≤ OPT + $1/3 \cdot$OPT ≤ $r_m \cdot$OPT

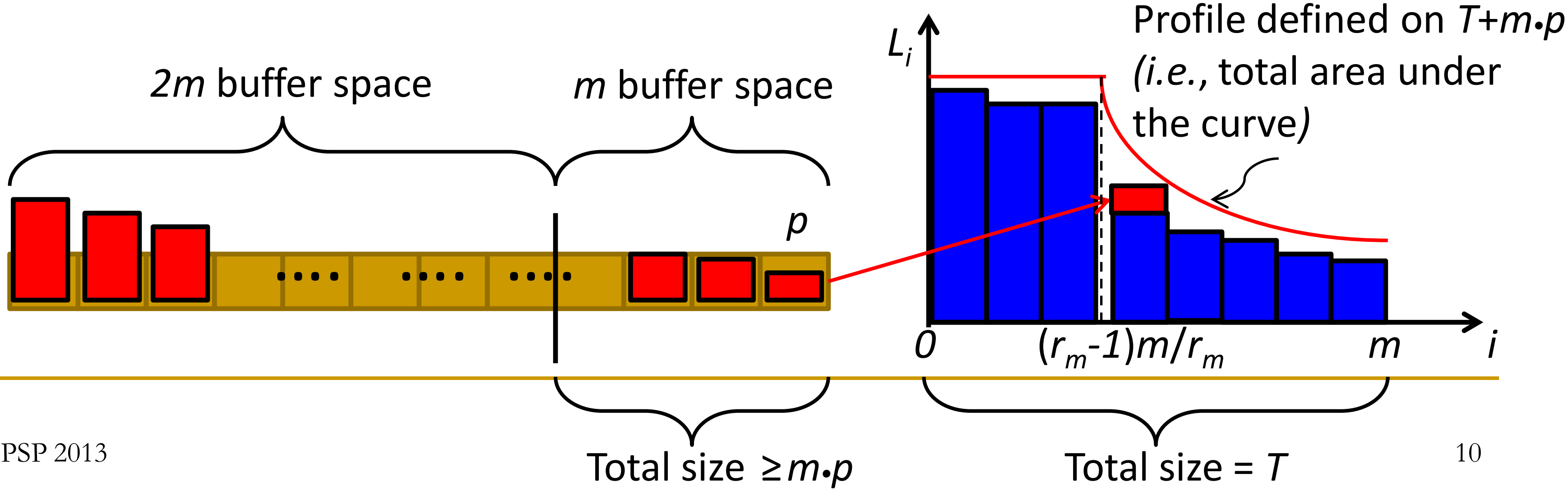
    → Still optimal comp. ratio





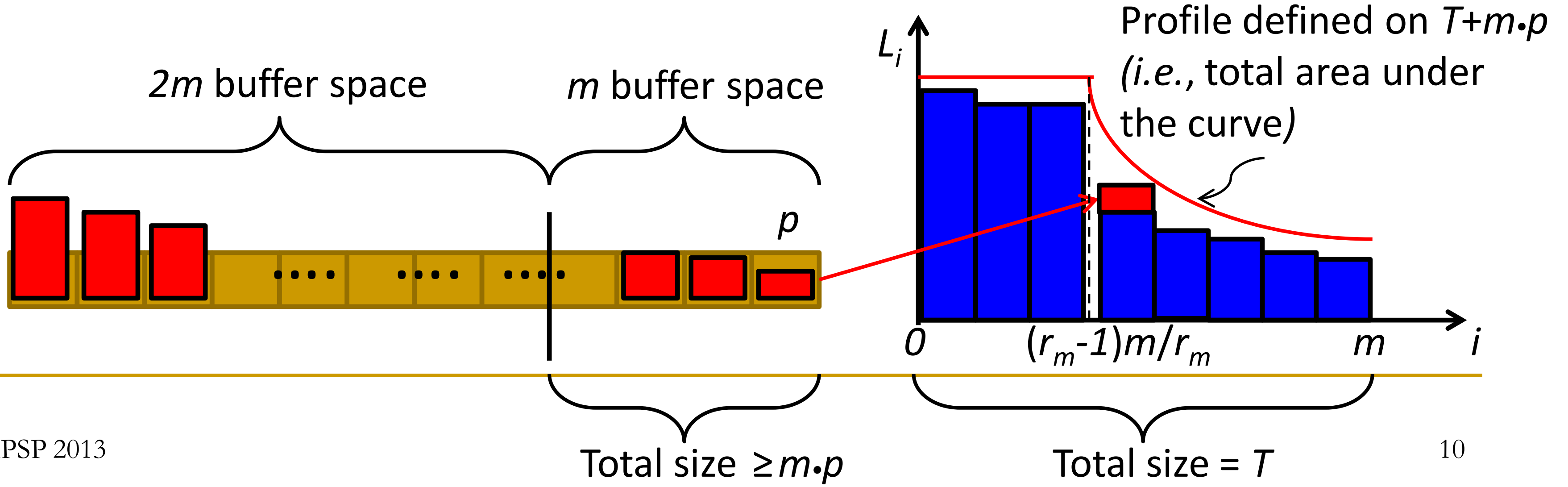
Buffer size ***k* = *3m*** (Englert et al.)

- **Iterative phase**: Assign smallest job of size p to a machine $\boldsymbol{j}$ with load $\boldsymbol{L_j \leq w_j \cdot (T+m \cdot p) - p}$



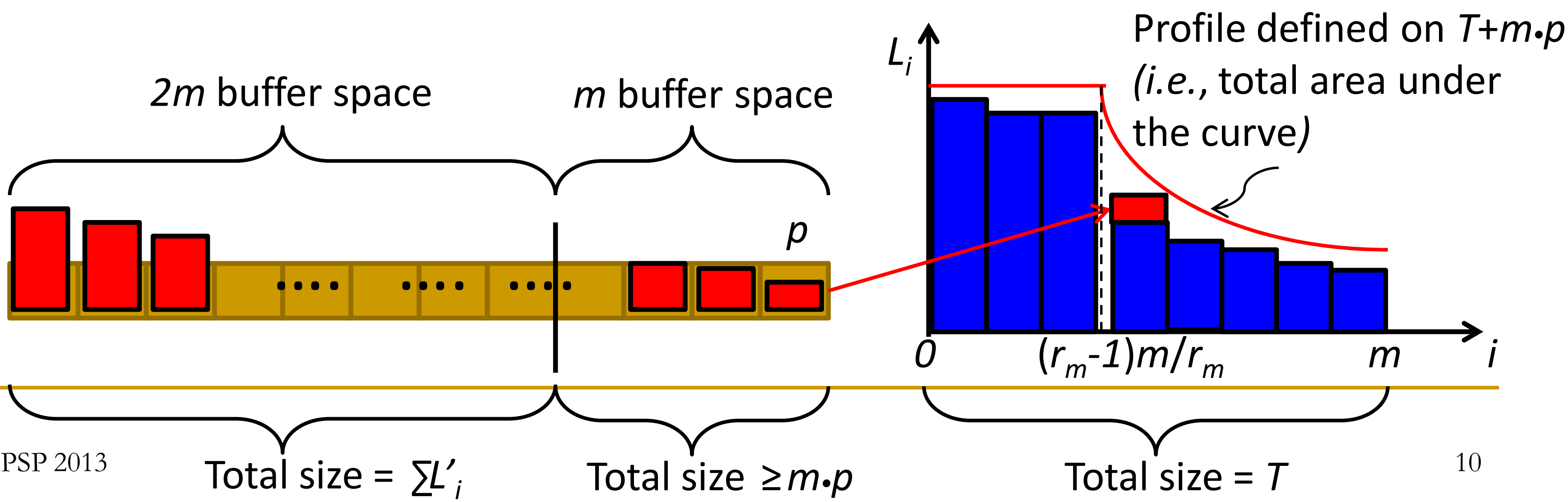
Buffer size ***k = 3m*** (Englert et al.)

- **Iterative phase**: Assign smallest job of size p to a machine $\boldsymbol{j}$ with load $\boldsymbol{L_j \leq w_j \cdot (T+m \cdot p) - p}$
  - This is feasible with at least m buffer space *(proof by contradiction)*
  - **After Iterative phase**: no machine exceeds the profile defined on $T_{\text{final}}+m \cdot p$



Buffer size $k = 3m$ (Englert et al.)

- **Iterative phase**: Assign smallest job of size p to a machine j with load $L_j \leq w_j \cdot (T+m \cdot p) - p$
  - This is feasible with at least m buffer space *(proof by contradiction)*
  - **After Iterative phase**: no machine exceeds the profile defined on $T_{\text{final}}+m \cdot p$
  - **Final phase**: at most $2m$ **large** job form an optimal schedule $L'_1 \leq L'_2 \leq \ldots \leq L'_m$

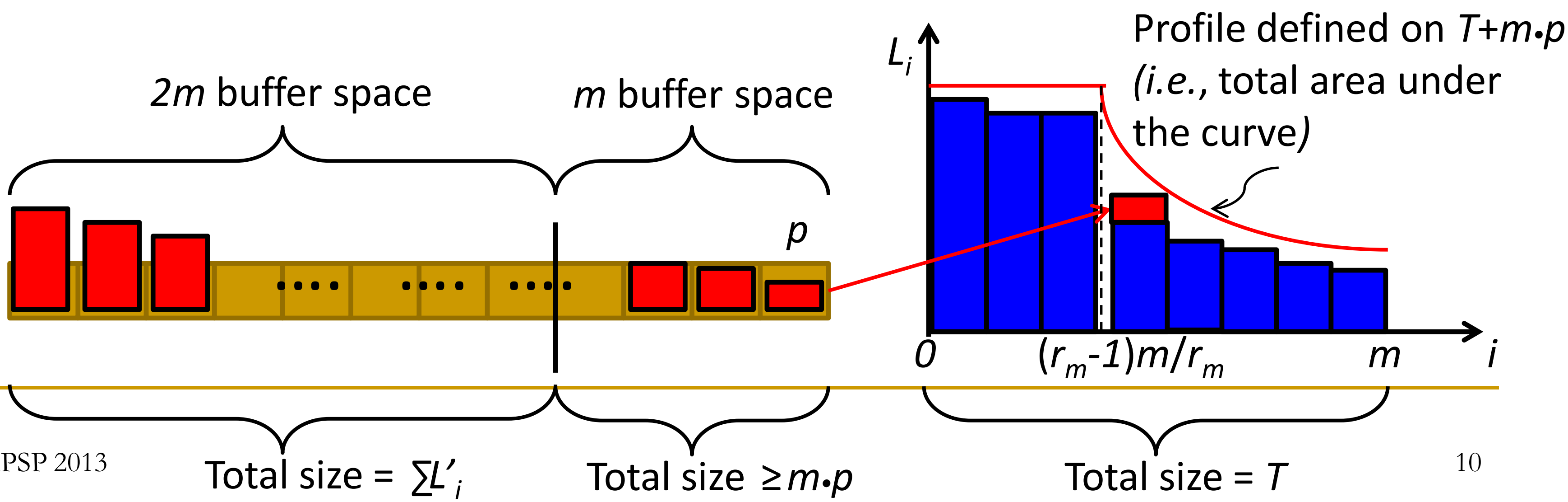


Buffer size $k = 3m$ (Englert et al.)

- **Iterative phase**: Assign smallest job of size p to a machine j with load $L_j \le w_j \cdot (T + m \cdot p) - p$
  - This is feasible with at least m buffer space *(proof by contradiction)*
  - **After Iterative phase**: no machine exceeds the profile defined on $T_{\text{final}} + m \cdot p$
  - **Final phase**: at most $2m$ **large** job form an optimal schedule $L'_1 \le L'_2 \le \ldots \le L'_m$

For all $0 \le j \le m-1$

$$\left.\begin{aligned} OPT &\ge (T_{\text{final}} + m \cdot p + \textstyle\sum L'_i)/m \\ L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \end{aligned}\right\} \Rightarrow L_j \le r_m \cdot OPT$$



Buffer size $k = 3m$ (Englert et al.)

***Mathematics to prove the 1st step of the final phase*:**

For all $0 \le j \le m-1$

$$\left.\begin{array}{l} OPT \ge (T_{\text{final}} + m \cdot p + \sum L'_i)/m \\ L_j \le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \end{array}\right\} \Rightarrow L_j \le r_m \cdot OPT$$

- For $(r_m-1)m/r_m \le j \le m-1$, $w_j = (r_m-1)/j$

$$\begin{aligned} L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - \textstyle\sum L'_i) + L'_j \\ &\le (r_m-1)/j \cdot (m\,OPT - (m-j)\,L'_j) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - m\,L'_j + j\,L'_j) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - m\,L'_j) + r_m\,L'_j \\ &\le r_m/m \cdot (m\,OPT - m\,L'_j) + r_m\,L'_j \\ &= r_m \cdot OPT \end{aligned}$$

- For $0 \le j \le (r_m-1)m/r_m$, $w_j = r_m/m$

$$\begin{aligned} L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \\ &\le r_m/m \cdot (m\,OPT - \textstyle\sum L'_i) + L'_j \\ &\le r_m/m \cdot (m\,OPT - (m-j)\,L'_j) + L'_j \\ &= r_m/m \cdot (m\,OPT - m/r_m\,L'_j) + L'_j \\ &= r_m \cdot OPT \end{aligned}$$

Buffer size ***k = 3m*** (Englert et al.)

***Mathematics to prove the 1st step of the final phase*:**

For all $0 \le j \le m-1$

$$\left.\begin{aligned} OPT &\ge (T_{\text{final}} + m \cdot p + \sum L'_i)/m \\ L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \end{aligned}\right\} \Rightarrow L_j \le r_m \cdot OPT$$

- **For** $(r_m-1)m/r_m \le j \le m-1$, $w_j = (r_m-1)/j$

$$\begin{aligned} L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - \textstyle\sum L'_i) + L'_j \\ &\le (r_m-1)/j \cdot (m\,OPT - (m-j)\,L'_j) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - m\,L'_j + j\,L'_j) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - m\,L'_j) + r_m\,L'_j \\ &\le r_m/m \cdot (m\,OPT - m\,L'_j) + r_m\,L'_j \\ &= r_m \cdot OPT \end{aligned}$$

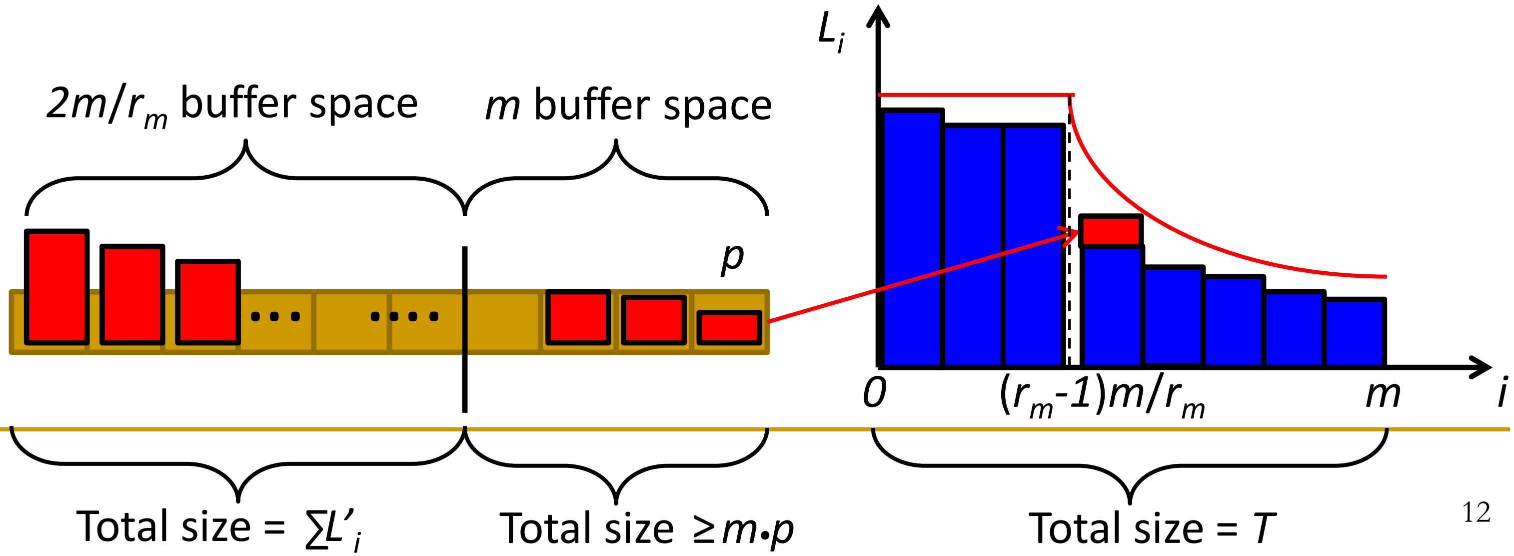
- **For** $0 \le j \le (r_m-1)m/r_m$, $w_j = r_m/m$

$$\begin{aligned} L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \\ &\le r_m/m \cdot (m\,OPT - \textstyle\sum L'_i) + L'_j \\ &\le r_m/m \cdot (m\,OPT - (m-j)\,L'_j) + L'_j \\ &= r_m/m \cdot (m\,OPT - m/r_m\,L'_j) + L'_j \\ &= r_m \cdot OPT \end{aligned}$$

Similar derivations carry over to the other algorithms

Buffer size $k = (1+2/r_m)m$ (Englert et al.)

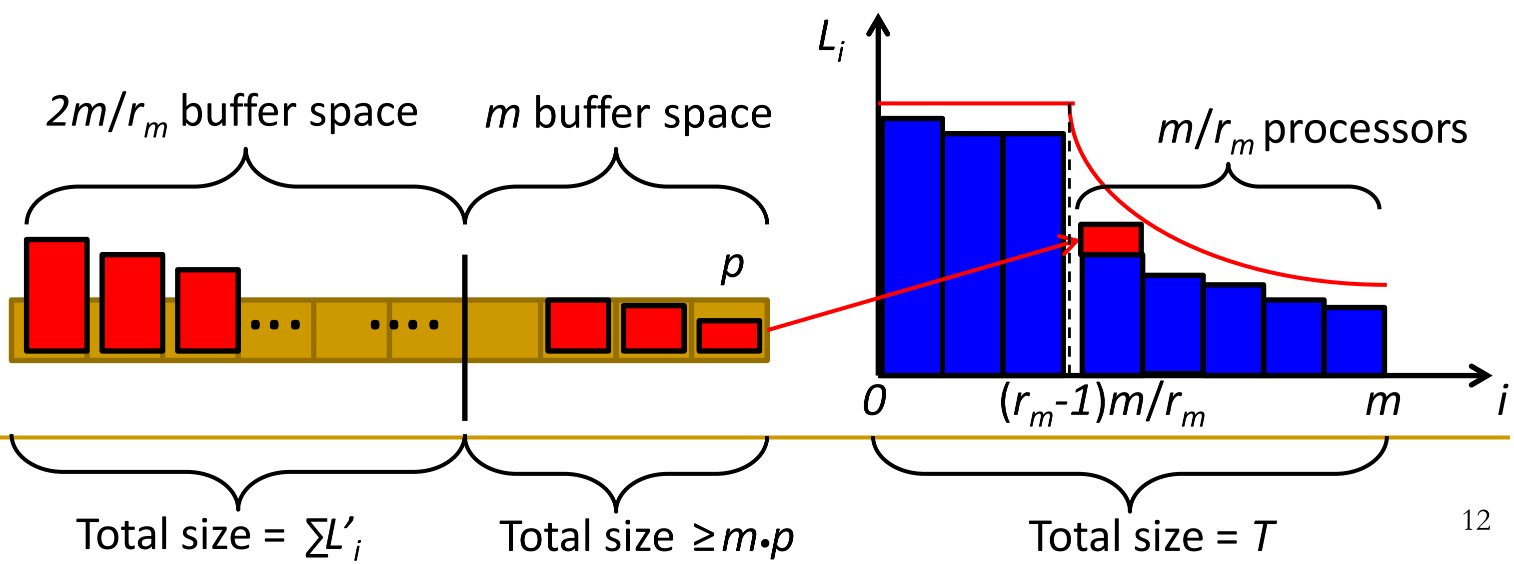
- **Same algorithm:** Requires a buffer size ≈ $2.364m$ for large m



Buffer size $\boldsymbol{k = (1+2/r_m)m}$ (Englert et al.)

- **Same algorithm:** Requires a buffer size ≈ $2.364m$ for large m
  - **Observation**: The following needs only hold for the m/r_m processors on the right, since **the profile on the left is flat**

$\boxed{\textbf{For } (r_m\text{-}1)m/r_m \le j \le m\text{-}1}$ $\left.\begin{array}{r} OPT \ge (T_{\text{final}} + m{\cdot}p + \sum L'_i)/m \\ L_j \le w_j \cdot (T_{\text{final}} + m{\cdot}p) + L'_j \end{array}\right\} \Rightarrow L_j \le r_m \cdot OPT$



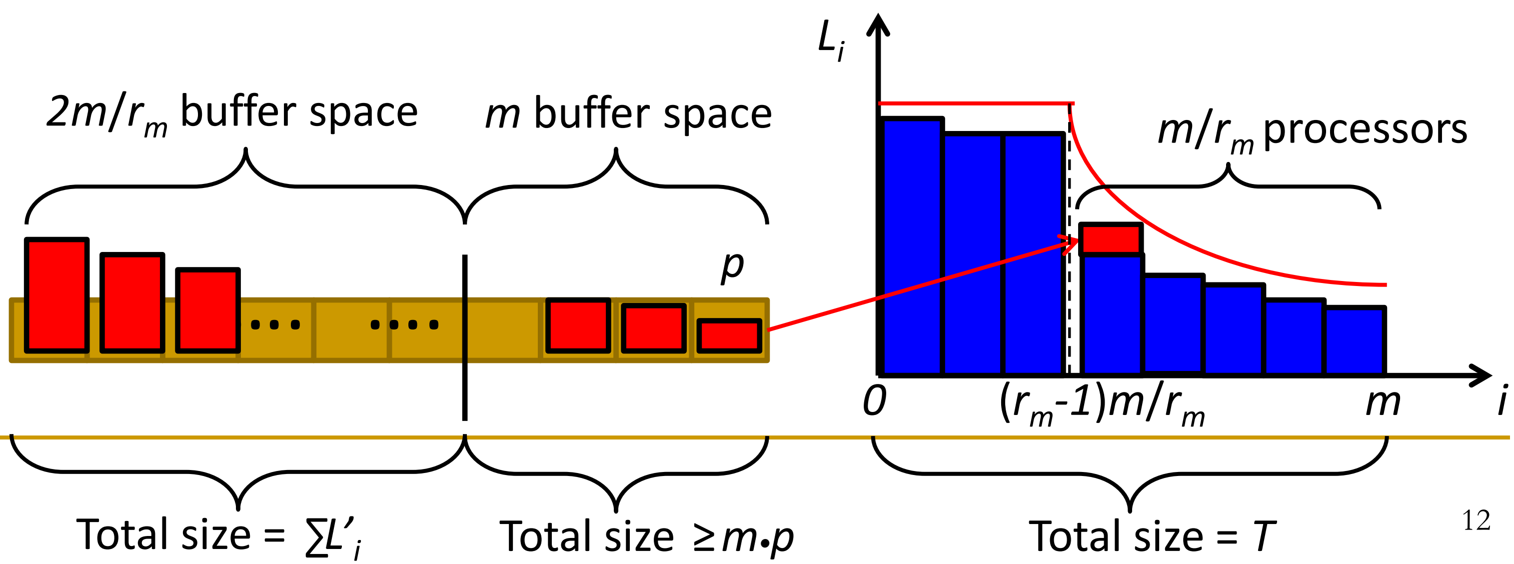
Buffer size $k = (1+2/r_m)m$ (Englert et al.)

- **Same algorithm:** Requires a buffer size ≈ *2.364m* for large *m*
  - **Observation**: The following needs only hold for the m/r_m processors on the right, since **the profile on the left is flat**

$\boxed{\textbf{For } (r_m\text{-}1)m/r_m \le j \le m\text{-}1}$

$$\left.\begin{array}{r} OPT \ge (T_{\text{final}} + m{\bullet}p + \sum L'_i)/m \\ L_j \le w_j \cdot (T_{\text{final}} + m{\bullet}p) + L'_j \end{array}\right\} \Rightarrow L_j \le r_m \cdot OPT$$

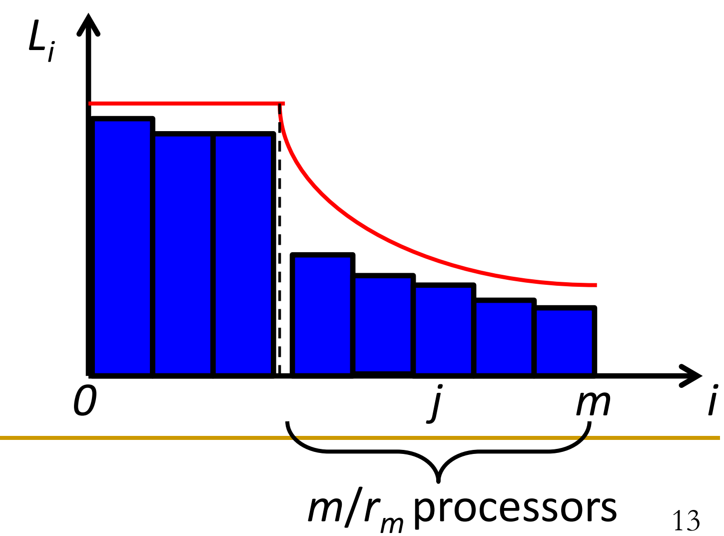
These processors get at most *2* jobs each according to the optimal LPT rule, so total extra buffer space required is $2m/r_m$



Buffer size $k = (5\text{-}2r_m)m$ (Our Result)

- **Observation**: In the *iterative phase*, it is **not** necessary to maintain a **uniform** load profile for all processors, or more precisely for the m/r_m processors on the right, in order to satisfy the following in the **final phase**

$$\boxed{\text{For } (r_m-1)m/r_m \le j \le m-1} \quad \left.\begin{array}{r} OPT \ge (T_{\text{final}} + m \cdot p + \sum L'_i)/m \\ L_j \le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \end{array}\right\} \Rightarrow L_j \le r_m \cdot OPT$$



Buffer size $\boldsymbol{k = (5\text{-}2r_m)m}$ (Our Result)

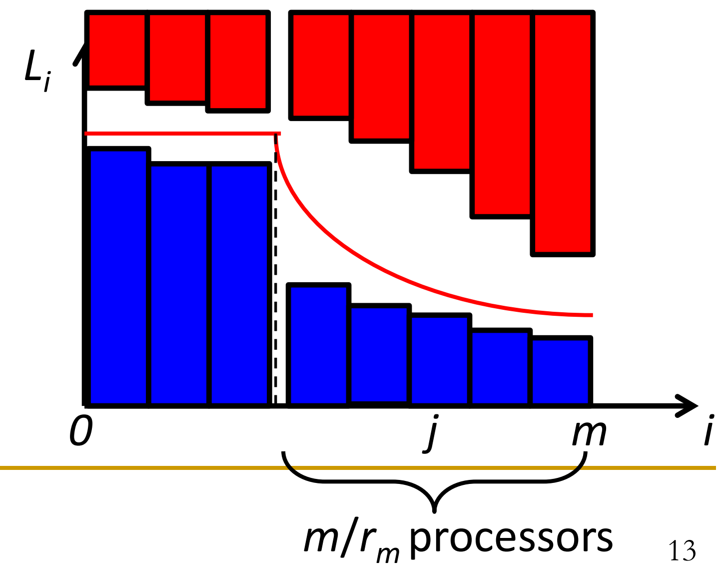
- **Observation**: In the *iterative phase*, it is **not** necessary to maintain a **uniform** load profile for all processors, or more precisely for the m/r_m processors on the right, in order to satisfy the following in the **final phase**

$$\boxed{\text{For } (r_m-1)m/r_m \le j \le m-1} \quad \left.\begin{array}{r} OPT \ge (T_{\text{final}} + m \cdot p + \sum L'_i)/m \\ L_j \le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \end{array}\right\} \Rightarrow L_j \le r_m \cdot OPT$$

- **Design a non-uniform profile**: Observe from the proof of the final phase

For $(r_m-1)m/r_m \le j \le m-1$, $w_j = (r_m-1)/j$

$$\begin{aligned}
L_j &\le w_j \cdot (T_{\text{final}} + m \cdot p) + L'_j \\
&= (r_m-1)/j \cdot (m\, OPT - \textstyle\sum L'_i) + L'_j \\
&\le (r_m-1)/j \cdot (m\, OPT - (m-j)\, L'_j) + L'_j \\
&= (r_m-1)/j \cdot (m\, OPT - m\, L'_j + j\, L'_j) + L'_j \\
&= (r_m-1)/j \cdot (m\, OPT - m\, L'_j) + r_m\, L'_j \\
&\le r_m/m \cdot (m\, OPT - m\, L'_j) + r_m\, L'_j \\
&= r_m \cdot OPT
\end{aligned}$$



Buffer size $k = (5-2r_m)m$ (Our Result)

- **Observation**: In the *iterative phase*, it is **not** necessary to maintain a **uniform** load profile for all processors, or more precisely for the m/r_m processors on the right, in order to satisfy the following in the **final phase**

$\boxed{\text{For } (r_m-1)m/r_m \le j \le m-1}$

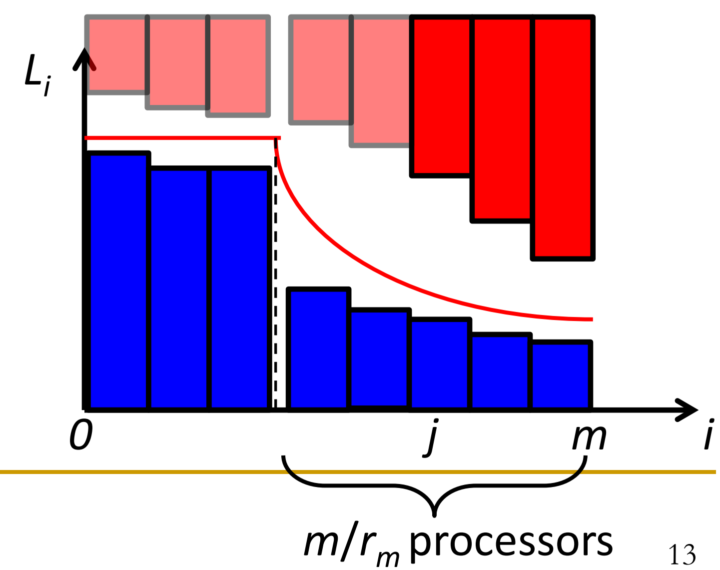
$$\left.\begin{aligned} OPT &\ge (T_{\text{final}} + m{\cdot}p + \textstyle\sum L'_i)/m \\ L_j &\le w_j \cdot (T_{\text{final}} + m{\cdot}p) + L'_j \end{aligned}\right\} \Rightarrow L_j \le r_m \cdot OPT$$

- **Design a non-uniform profile**: Observe from the proof of the final phase

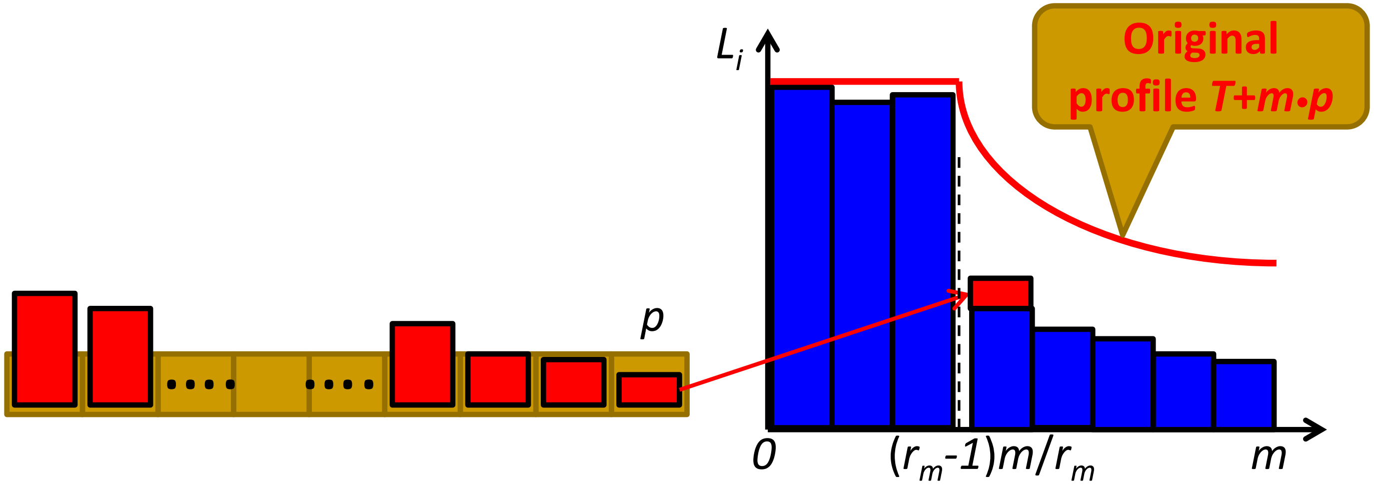
For $(r_m-1)m/r_m \le j \le m-1$, $w_j = (r_m-1)/j$

$$\begin{aligned} L_j &\le w_j \cdot (T_{\text{final}} + m{\cdot}p) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - \textstyle\sum L'_i) + L'_j \\ &\le (r_m-1)/j \cdot (m\,OPT - (m-j)\,L'_j) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - m\,L'_j + j\,L'_j) + L'_j \\ &= (r_m-1)/j \cdot (m\,OPT - m\,L'_j) + r_m\,L'_j \\ &\le r_m/m \cdot (m\,OPT - m\,L'_j) + r_m\,L'_j \\ &= r_m \cdot OPT \end{aligned}$$

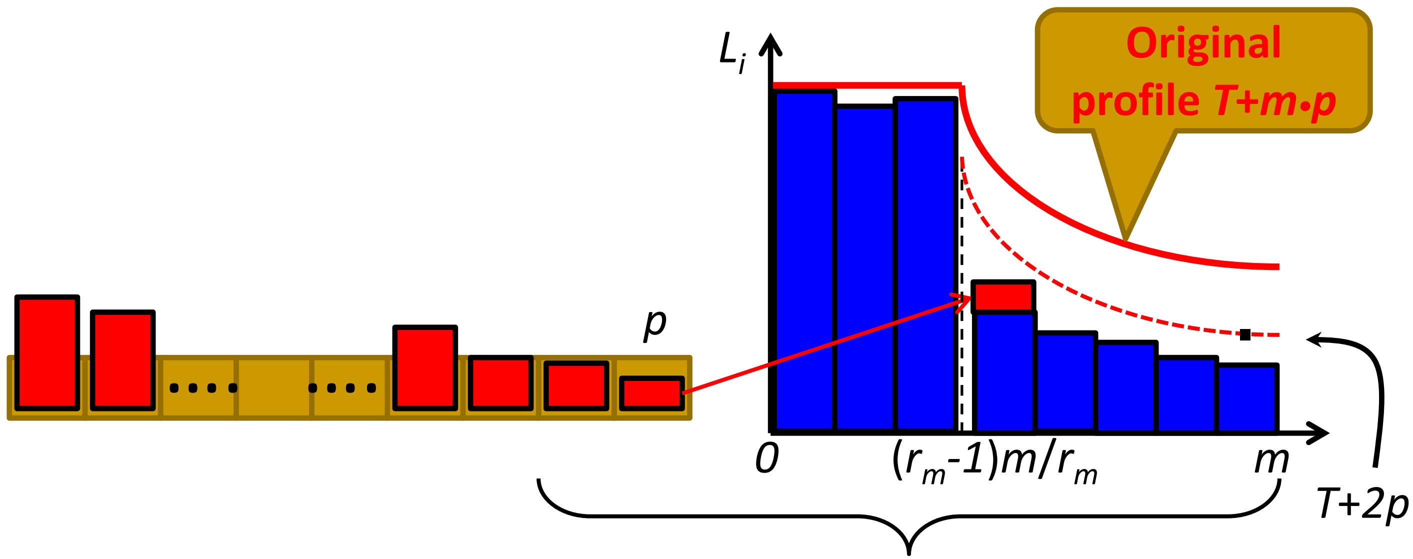
$\boxed{\textstyle\sum L'_i \ge (m-j)L'_j,\ \text{giving up } L'_1 \text{ to } L'_{j-1}}$



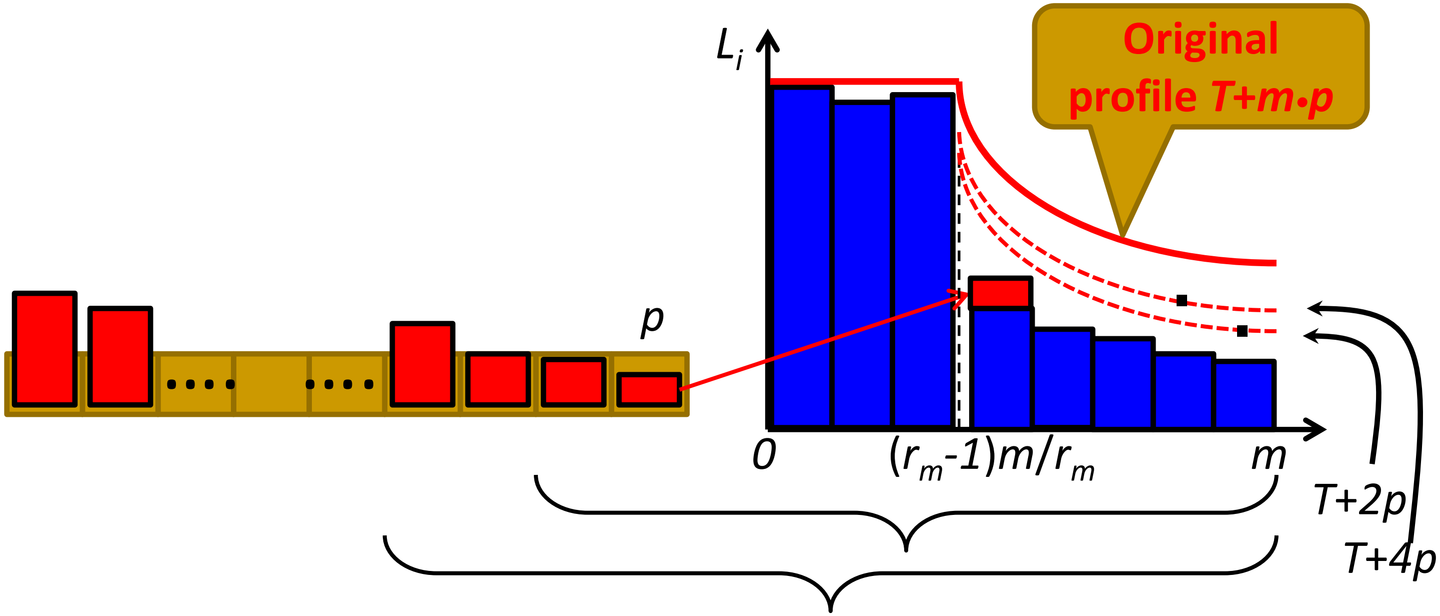
Buffer size $k = (5-2r_m)m$ (Our Result)



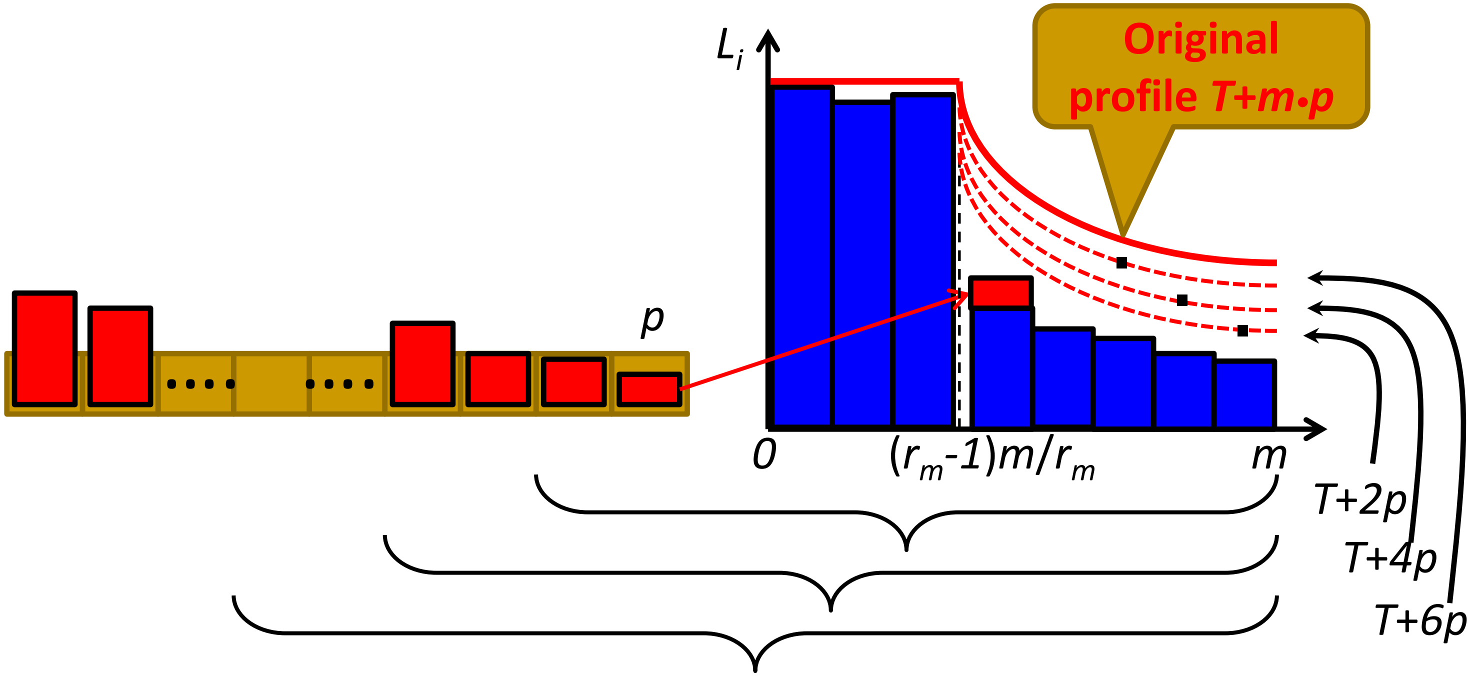
Buffer size $k = (5\text{-}2r_m)m$ (Our Result)



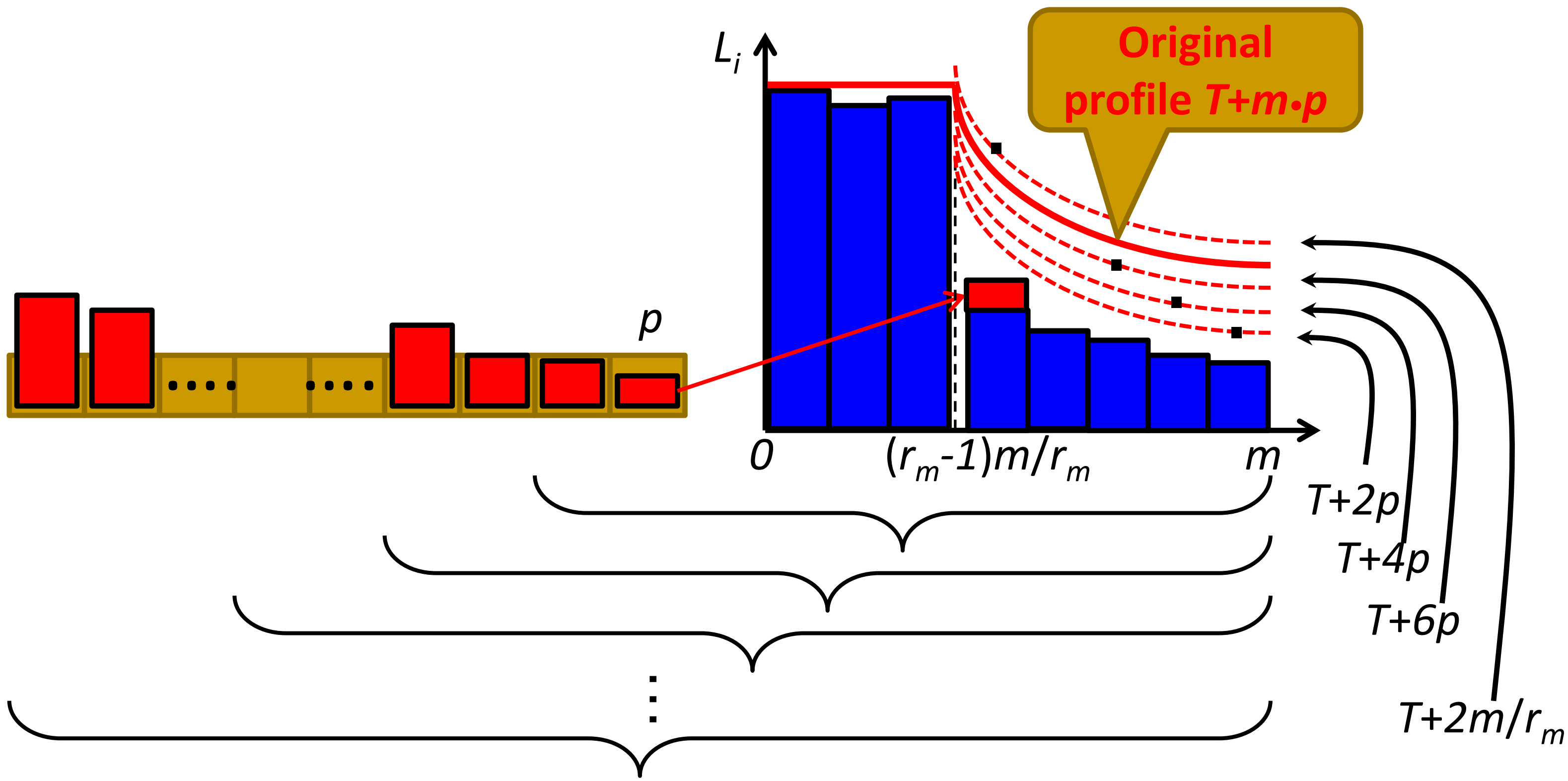
Buffer size $k = (5\text{-}2r_m)m$ (Our Result)



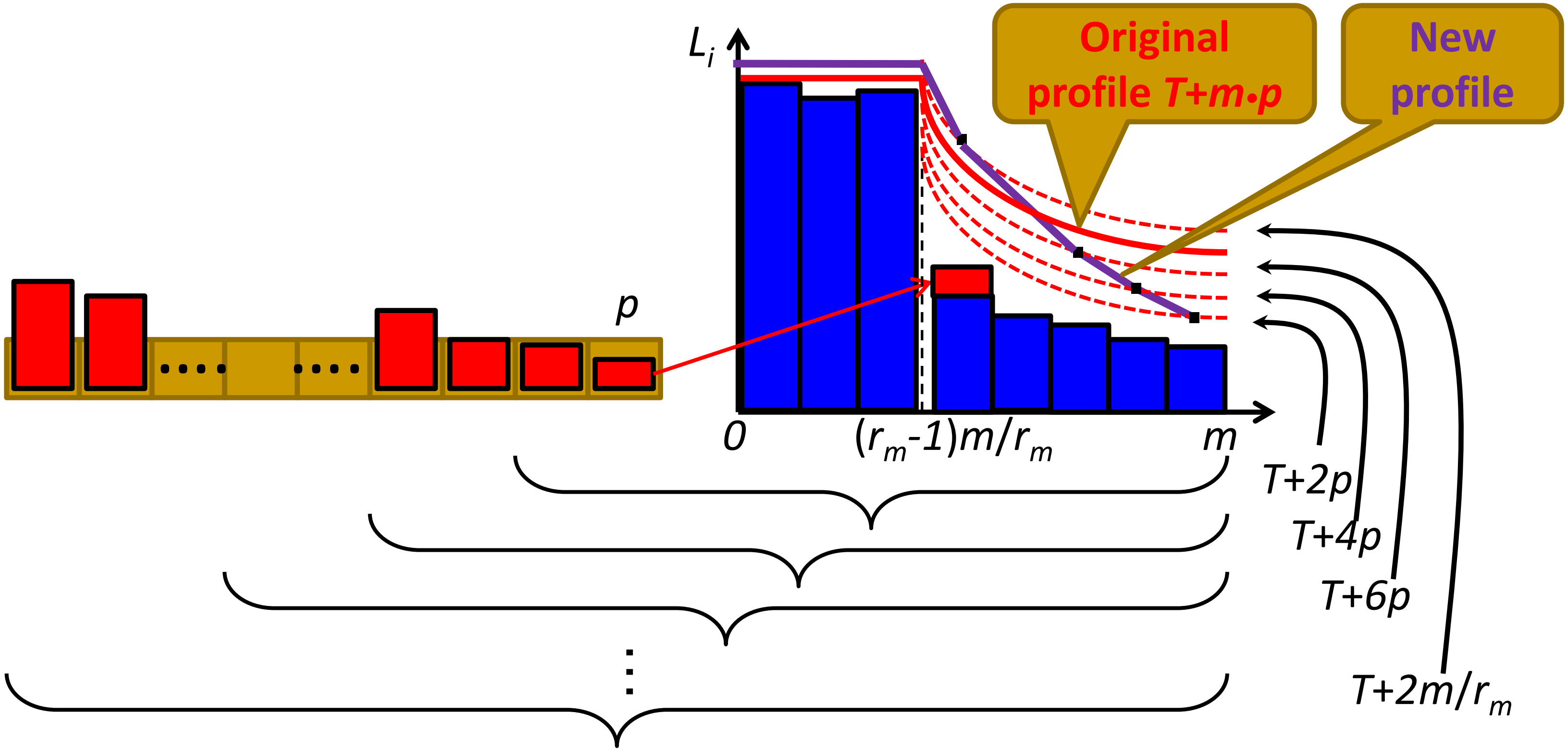
Buffer size $k = (5\text{-}2r_m)m$ (Our Result)



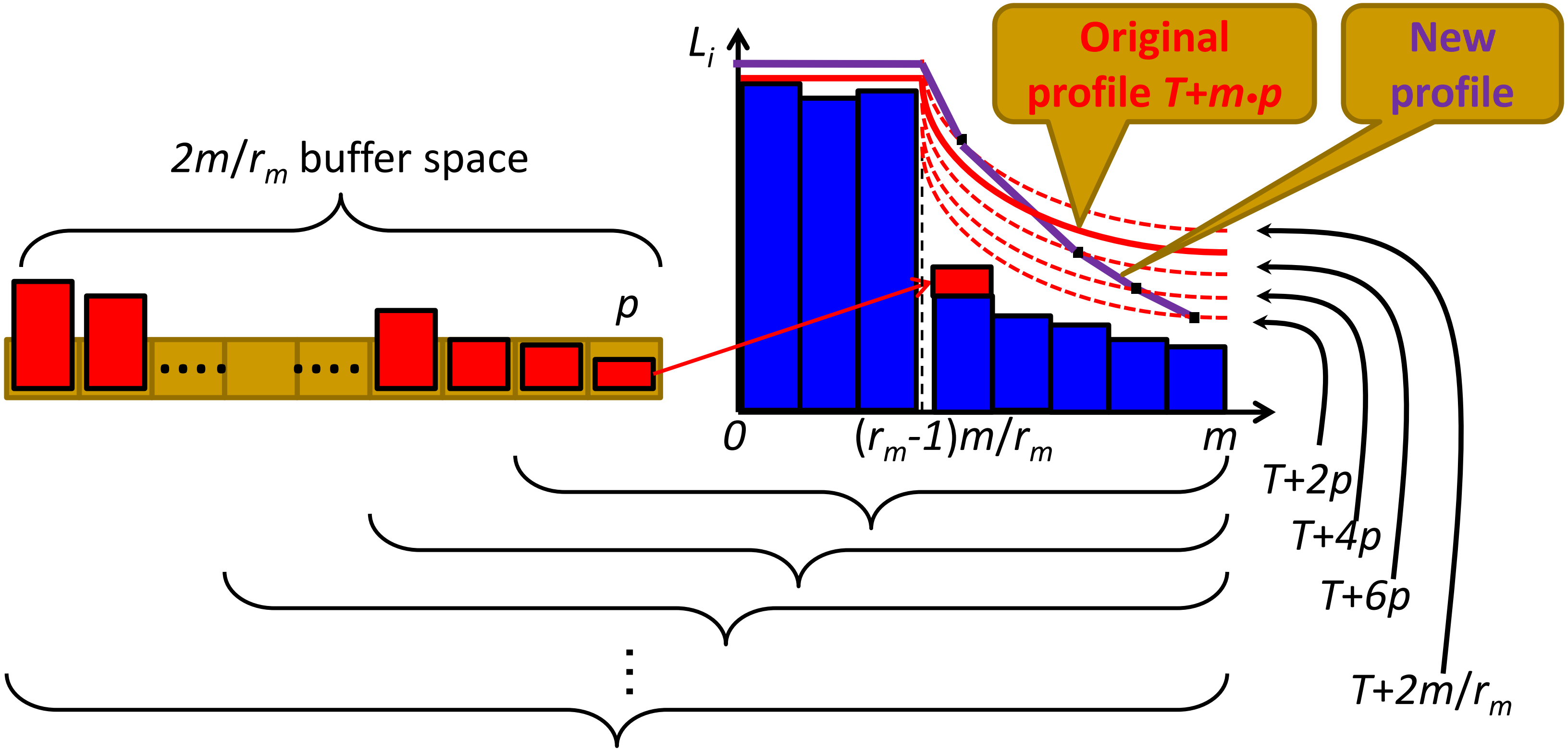
Buffer size $k = (5\text{-}2r_m)m$ (Our Result)



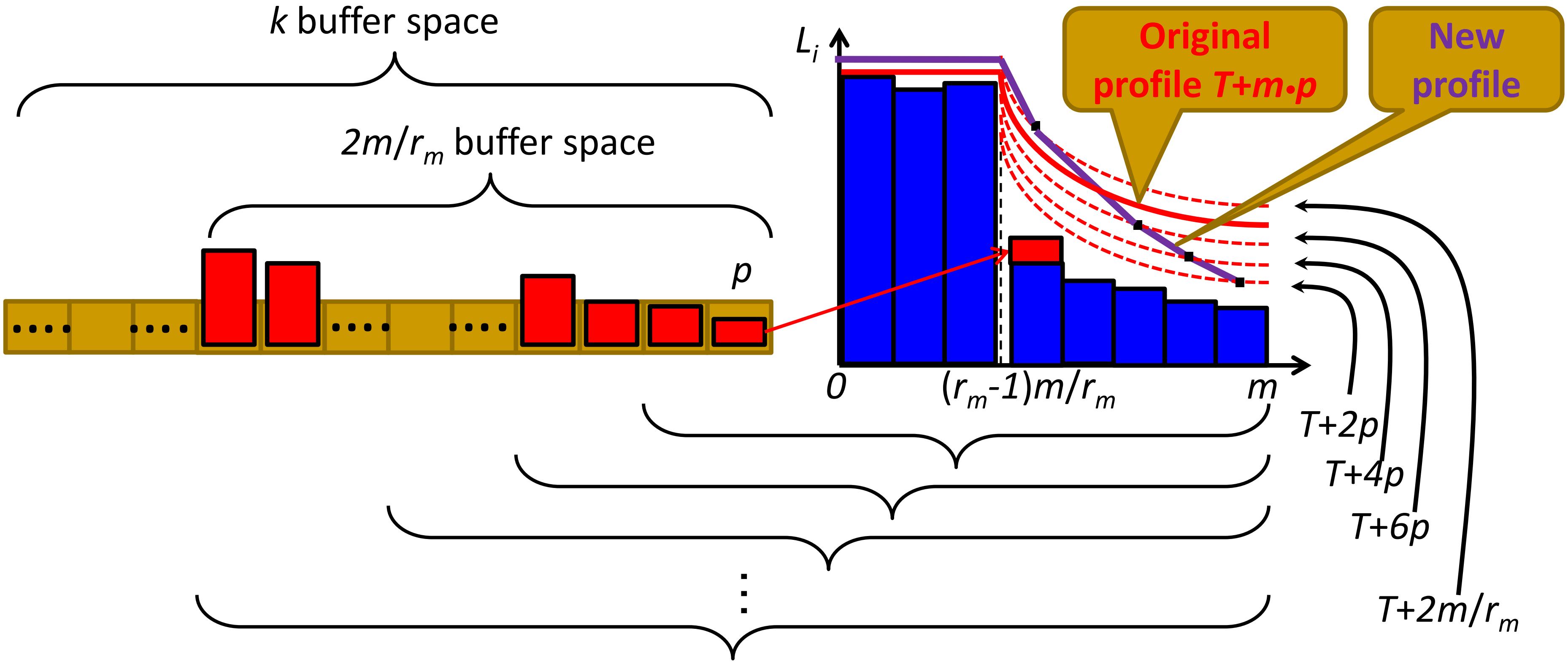
Buffer size $k = (5\text{-}2r_m)m$ (Our Result)



Buffer size $k = (5\text{-}2r_m)m$ (Our Result)

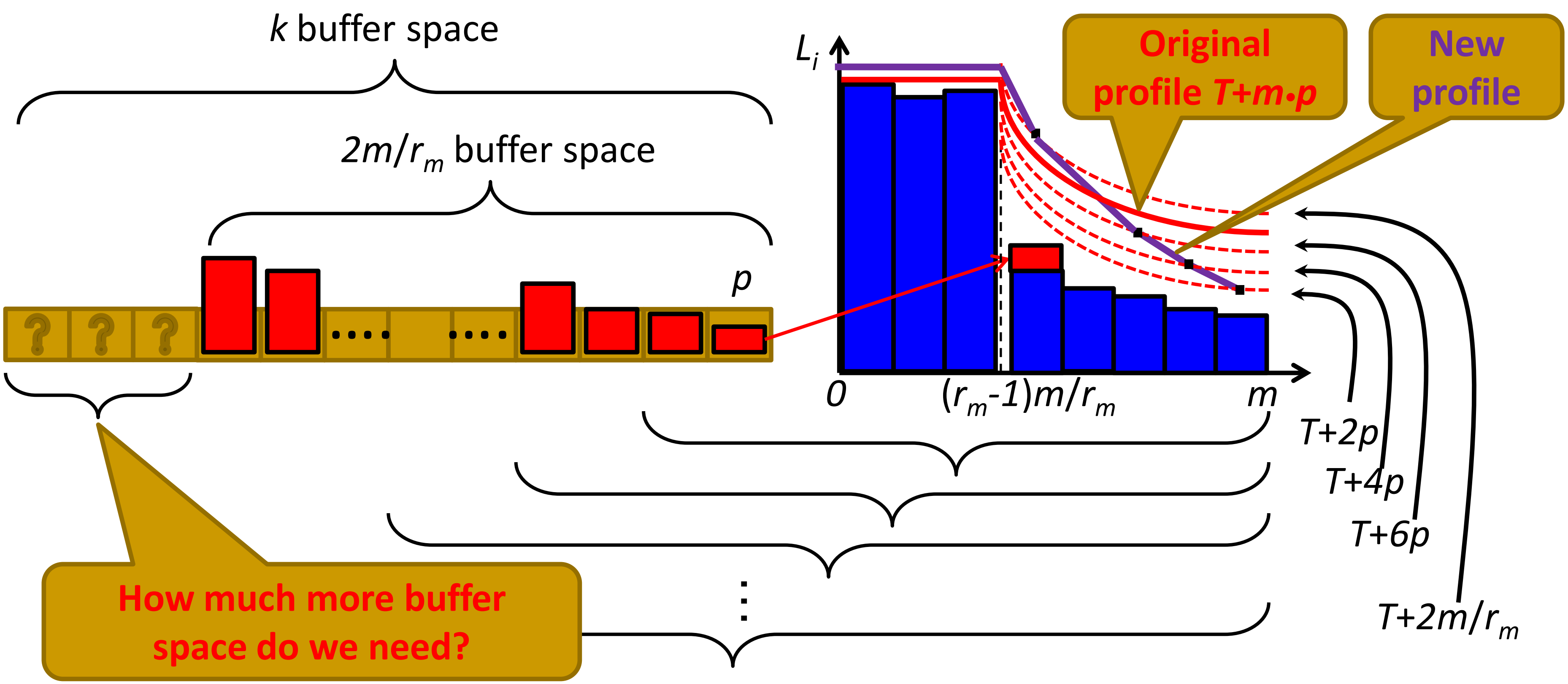


Buffer size $k = (5-2r_m)m$ (Our Result)



Iterative phase: Assign smallest job of size p to a machine j with load $\leq w_j \cdot (T+k-2(m-j')p) - p$, where $j' = \max\{j, (r_m-1)m/r_m\}$

Buffer size $k = (5\text{-}2r_m)m$ (Our Result)



Iterative phase: Assign smallest job of size p to a machine j with load $\leq w_j \cdot (T+k-2(m-j')p) - p$, where $j' = \max\{j, (r_m-1)m/r_m\}$

Buffer size $k = (5-2r_m)m$ (Our Result)

- **Determine total buffer size *k*:** Apply the ***feasibility condition*** in the ***iterative phase***
  - At any time, there exists a machine j with load $\leq w_j \cdot (T+k-2(m-j')p) - p$, where $j' = \max\{j, (r_m-1)m/r_m\}$

Buffer size $k = (5\text{-}2r_m)m$ (Our Result)

- **Determine total buffer size *k*:** Apply the ***feasibility condition*** in the ***iterative phase***
  - At any time, there exists a machine j with load $\le w_j \cdot (T+k-2(m-j')p) - p$, where $j' = \max\{j, (r_m-1)m/r_m\}$
  - As shown previously, at least $m \cdot p$ space between the current load T and the designed profile will suffice

$$\sum w_j \cdot (T+k-2(m-j')p) \ge T + m \cdot p$$

$$\Leftarrow T + k - 2m \cdot p + 2\sum w_j \cdot j' \ge T + m \cdot p \qquad (\textstyle\sum w_j = 1)$$

$$\Leftarrow T + k - 2m \cdot p + 2(r_m-1)m \ge T + m \cdot p \qquad (\textstyle\sum w_j \cdot j' \ge (r_m-1)m)$$

$$\Leftarrow k \ge (5-2r_m)m$$

Buffer size $k = (5\text{-}2r_m)m$ (Our Result)

- **Determine total buffer size *k*:** Apply the ***feasibility condition*** in the ***iterative phase***
  - At any time, there exists a machine j with load $\le w_j \cdot (T+k-2(m-j')p) - p$, where $j' = \max\{j, (r_m-1)m/r_m\}$
  - As shown previously, at least $m \cdot p$ space between the current load T and the designed profile will suffice

$$\sum w_j \cdot (T+k-2(m-j')p) \ge T + m \cdot p$$

$$\leftarrow T + k - 2m \cdot p + 2\sum w_j \cdot j' \ge T + m \cdot p \qquad (\sum w_j = 1)$$

$$\leftarrow T + k - 2m \cdot p + 2(r_m-1)m \ge T + m \cdot p \qquad (\sum w_j \cdot j' \ge (r_m-1)m)$$

$$\leftarrow k \ge (5-2r_m)m$$

m	*2*	*3*	*4*	…	*10*	*20*	*30*	…	*m→∞*
Old *k*	*6*	*9*	*11*	…	*2.5m*	*2.455m*	*2.406m*	…	*2.364m*
New *k*	*6*	*8*	*10*	…	*2.25m*	*2.182m*	*2.125m*	…	*2.068m*

Buffer size $k = (5-2r_m)m$ (Our Result)

- **Determine total buffer size k:** Apply the ***feasibility condition*** in the ***iterative phase***
  - At any time, there exists a machine j with load $\leq w_j \cdot (T+k-2(m-j')p) - p$, where $j' = \max\{j, (r_m-1)m/r_m\}$
  - As shown previously, at least $m \cdot p$ space between the current load T and the designed profile will suffice

$$\sum w_j \cdot (T+k-2(m-j')p) \geq T + m \cdot p$$

$$\Leftarrow T + k - 2m \cdot p + 2\sum w_j \cdot j' \geq T + m \cdot p \qquad (\textstyle\sum w_j = 1)$$

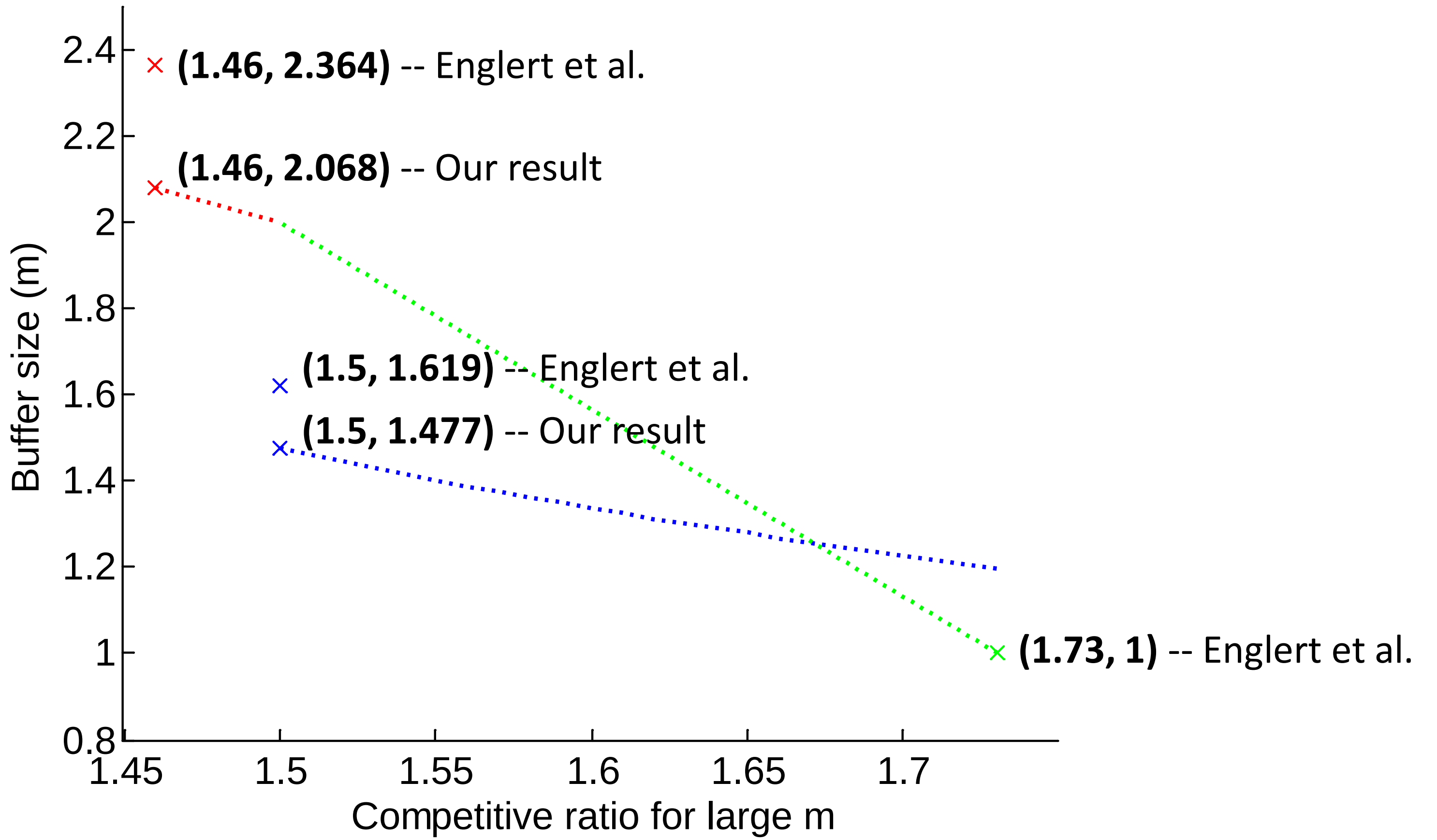
$$\Leftarrow T + k - 2m \cdot p + 2(r_m-1)m \geq T + m \cdot p \qquad (\textstyle\sum w_j \cdot j' \geq (r_m-1)m)$$

$$\Leftarrow k \geq (5-2r_m)m$$

m	2	3	4	…	10	20	30	…	$m \to \infty$
Old k	6	9	11	…	2.5m	2.455m	2.406m	…	2.364m
New k	6	8	10	…	2.25m	2.182m	2.125m	…	2.068m
	1	6							

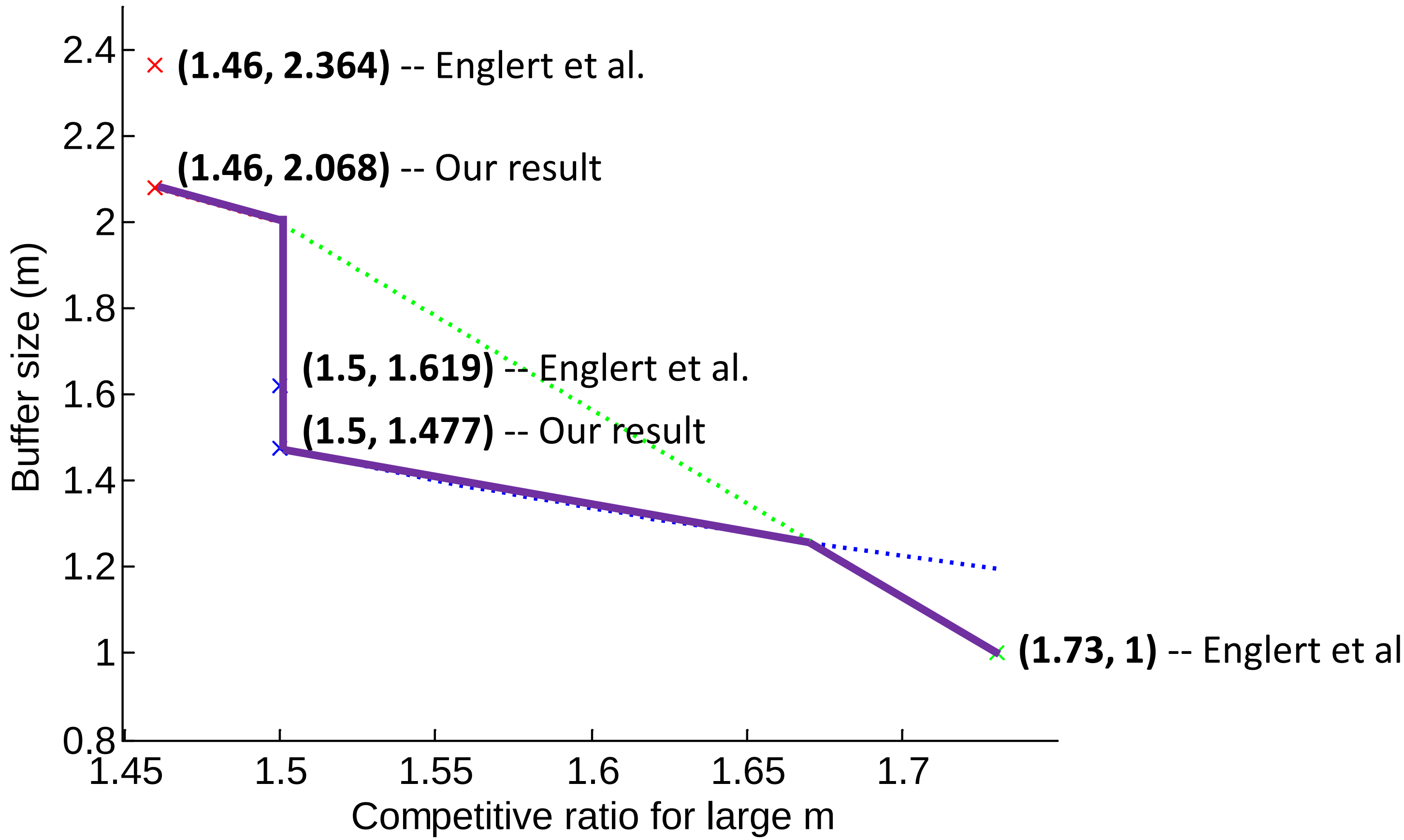
⇐ Lan et al. 2012

Tradeoff: Comp. ratio vs Buffer size



Combining and extending our results with the ones from Englert et al. 2008

Tradeoff: Comp. ratio vs Buffer size



Combining and extending our results with the ones from Englert et al. 2008

Remarks

- **Classical online makespan scheduling**
  - **Comp. ratio for large *m***

Upper Bound:

2	(Graham 1966)
1.986	(Bartal et al. 1995)
1.945	(Karger et al. 1996)
1.923	(Albers 1999)
1.9201	(Fleischer et al. 2000)
….	

Lower Bound:

1.88	(Rudin III 2001)
1.854	(Gormley et al. 2000)
1.852	(Albers 1999)
1.837	(Bartal et al. 1994)

- **Semi-online scheduling with reordering buffer**
  - **Buffer size needed to get opt. comp. ratio for large *m***

2.364m	(Englert et al. 2008)
2.068m	(Our result)
….	
….	
m?	(Our work in progress)
0.5m	(Englert et al. 2008)

Remarks

- **Classical online makespan scheduling**
  - **Comp. ratio for large *m***

Upper Bound:

- *2* (Graham 1966)
- *1.986* (Bartal et al. 1995)
- *1.945* (Karger et al. 1996)
- *1.923* (Albers 1999)
- *1.9201* (Fleischer et al. 2000)
- ….

Lower Bound:

- *1.88* (Rudin III 2001)
- *1.854* (Gormley et al. 2000)
- *1.852* (Albers 1999)
- *1.837* (Bartal et al. 1994)

- **Semi-online scheduling with reordering buffer**
  - **Buffer size needed to get opt. comp. ratio for large *m***
    - *2.364m* (Englert et al. 2008)
    - *2.068m* (Our result)
    - ….
    - ….
    - *m*? (Our work in progress)
    - *0.5m* (Englert et al. 2008)

A similar problem

- **Semi-online scheduling with job migrations**
  - **Problem**
    - Online algorithm is allowed to perform a limited number (independent of input size) of job migrations
  - **Results** (Albers and Hellwig 2012)
    - For general m, optimal comp. ratio (lower bound) = r_m **(Identical to the case with reordering buffer)**
    - An algorithm that achieves r_m-comp. with $[(2-r_m)/(r_m-1)^2+4]m \approx 7m$ migrations for large m
  - Results with migration can be transformed into results with reordering buffer, but not vise versa

A similar problem

- **Semi-online scheduling with job migrations**
  - **Problem**
    - Online algorithm is allowed to perform a limited number (independent of input size) of job migrations
  - **Results** (Albers and Hellwig 2012)
    - For general m, optimal comp. ratio (lower bound) = r_m **(Identical to the case with reordering buffer)**
    - An algorithm that achieves r_m-comp. with $[(2-r_m)/(r_m-1)^2+4]m \approx 7m$ migrations for large m
  - Results with migration can be transformed into results with reordering buffer, but not vise versa
  - What is min. no. of migrations required? Can the existing result be improved to maintain the optimal comp. ratio?

Thanks for your attention!