

# ORDER SCHEDULING IN AN ENVIRONMENT WITH DEDICATED RESOURCES IN PARALLEL

JOSEPH Y-T. LEUNG<sup>1</sup>, HAIBING LI<sup>1</sup>, AND MICHAEL PINEDO<sup>2</sup>

<sup>1</sup>*Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA*

<sup>2</sup>*Stern School of Business, New York University, 40 West Fourth Street, New York, NY 10012, USA*

## PROOF OF LEMMA 3.3

For ease of presentation, we define the following notations that will be used throughout the proof:

$F_j^{(i)}$ : The length on machine  $i$  ( $i = 1, 2, 3$ ) after the  $j^{\text{th}}$  order is scheduled.

$C_j^{(i)}$ : The finish time on machine  $i$  ( $i = 1, 2, 3$ ) of the  $j^{\text{th}}$  order.

$C_j = \max_{1 \leq i \leq 3} \{C_j^{(i)}\}$ : The finish time of the  $j^{\text{th}}$  order in the schedule.

The above notation is defined in terms of the position of an order. In contrast, the following notation is defined in terms of the order itself:

$C_{J_j}^{(i)}$ : The finish time on machine  $i$  ( $i = 1, 2, 3$ ) of order  $J_j$ .

$C_{J_j} = \max_{1 \leq i \leq 3} \{C_{J_j}^{(i)}\}$ : The finish time of order  $J_j$  in the schedule.

In addition, for convenience of describing a pattern of a schedule, we define:

$(P)^k$ : A pattern produced by repeating  $k$  times the subpattern  $P$ , which is a sequence of  $a$ ,  $b$  and  $c$ .

$(P_1/P_2)^k$ : A pattern produced by repeating  $k$  times the subpattern  $P_1$  or  $P_2$ , each of which is a sequence of  $a$ ,  $b$  and  $c$ .

We also define  $S_{OPT}$  to be an optimal schedule and:

$\sum C_j(S)$ : The total completion time of schedule  $S$ .

Finally, for convenience, for each order constructed from  $a_j$  ( $b_j, c_j$ ), we simply call it order  $a_j$  (respectively,  $b_j, c_j$ ). Also, we assume that the subscripts of the orders in a schedule are labelled in such a way that,

$a_j$  ( $b_j$ , or  $c_j$ ) ( $j = 1, 2, \dots, n$ ): means that there are  $(j - 1)$  orders of type  $a$  scheduled before  $a_j$  (respectively,  $b_j$ , or  $c_j$ ).

*Observation 1.* In  $S_{OPT}$ :

- i) The orders of type  $c$  must be scheduled by a sequence satisfying the SPT rule;
- ii) The first order in the schedule cannot be of type  $c$ ;
- iii) The last order in the schedule must be of type  $c$ .

**Proof.** *i)* It follows by an interchange argument.

*ii)* Suppose that  $S_{OPT}$  starts with an order of type  $c$ . From *i)*, this order must be  $c_1$ . We let the first type- $a$  or type- $b$  order be in position  $(k + 1) > 1$ . We denote this order as  $c^*$ . Then, the first  $(k + 1)$  orders are scheduled as:

$$c_1 c_2 \dots c_k c^*.$$



The finish time of order  $c_j (j = 1, 2, \dots, k)$  in  $c_1 c_2 \dots c_k c^*$  is:

$$C_{c_j} = \max \left\{ \begin{array}{l} jL, \\ 2jL + 2 \sum_{l=1}^j c_l, \\ j(5L + 2X) + \sum_{l=1}^j c_l \end{array} \right\} = j(5L + 2X) + \sum_{l=1}^j c_l,$$

If we schedule  $c^*$  before  $c_1$  (i.e.,  $c^* c_1 c_2 \dots c_k$ ), it can be shown that the finish time of order  $c_j (j = 1, 2, \dots, k)$  remains the same as before. But the finish time of  $c^*$  in  $c^* c_1 c_2 \dots c_k$  is reduced by an amount of  $kL$  on machine 1 and an amount of  $2kL + 2 \sum_{l=1}^k c_l$  on machine 2. Therefore,  $c^* c_1 c_2 \dots c_k$  has smaller  $\sum C_j$  than  $c_1 c_2 \dots c_k c^*$ , contradicting the fact that  $S_{OPT}$  is optimal.

iii) Suppose that  $S_{OPT}$  does not end with the last type- $c$  order  $c_n$ . If we move order  $c_n$  to the end and push all orders scheduled after  $c_n$  forward, then the finish time of  $c_n$  in the new schedule is the same as the finish time of  $c_n$  in the old schedule. But the finish times of all orders that were pushed forward will decrease in the new schedule. Thus, the new schedule has a smaller  $\sum C_j$  than  $S_{OPT}$ , contradicting the fact that  $S_{OPT}$  is optimal.  $\blacksquare$

Observation 1 reveals the fact that it would be advantageous to distribute the remaining orders of type  $a$  and  $b$  within the  $n$  frames imposed by the orders of type  $c$ .

*Observation 2.* In  $S_{OPT}$  the first order must be of type  $a$ .

**Proof.** Let  $(k + 1) > 1$  be the smallest position of a type- $a$  order in  $S_{OPT}$ . Following our convention, we denote this type- $a$  order as  $a_1$ . We will show that there is a schedule  $S$  which has  $a_1$  in a position less than  $(k + 1)$ , such that

$$\sum_{j=1}^{3n} C_j(S) < \sum_{j=1}^{3n} C_j(S_{OPT}).$$

The first  $k$  orders are either of type- $b$  or type- $c$ . Let the number of type- $b$  and type- $c$  orders be  $n_b$  and  $n_c$ , respectively. Clearly,  $n_b + n_c = k$ . From Observation 1, the first order cannot be of type  $c$ . Thus,  $n_b \geq 1$ . We now show that  $n_c \geq 1$ .

Suppose  $n_c = 0$ . Consider the schedule  $S$  obtained from  $S_{OPT}$  by interchanging  $a_1$  with the order before it (which is  $b_k$ ). Then we have

$$\begin{aligned} & \sum C_j(S) - \sum C_j(S_{OPT}) = C_{a_1}(S) - C_{b_k}(S_{OPT}) \\ & = \max \left\{ \begin{array}{l} (k-1)(2L+2X) + \sum_{j=1}^{k-1} b_j + (2L+a_1), \\ (k-1)(L+X) - \sum_{j=1}^{k-1} b_j + (2L+X-a_1), \\ 0 \end{array} \right\} - \\ & \quad \max \left\{ \begin{array}{l} k(2L+2X) + \sum_{j=1}^k b_j, \\ k(L+X) - \sum_{j=1}^k b_j, \\ 0 \end{array} \right\} \\ & = -2X + a_1 - b_k < 0, \end{aligned}$$

contradicting the fact that  $S_{OPT}$  is optimal. Thus,  $n_c \geq 1$ .

Since  $k = n_b + n_c$ , we have  $k \geq 2$ . If  $k = 2$ , then the first three jobs of  $S_{OPT}$  must be  $b_1 c_1 a_1$ . Consider the schedule  $S$  obtained from  $S_{OPT}$  by swapping  $b_1$  with  $a_1$ . It is clear that the finish time of  $c_1$  in  $S$  remains unchanged. Thus,

$$\sum C_j(S) - \sum C_j(S_{OPT}) = C_{a_1}(S) - C_{b_1}(S_{OPT}) = -X - a_1 - b_1 < 0.$$

Therefore,  $S$  is better than  $S_{OPT}$ , contradicting the fact that  $S_{OPT}$  is optimal.

From now on, we will assume that  $k \geq 3$ . First, we have:

$$F_{k+1}^{(1)} = n_b(2L + 2X) + n_cL + \sum_{j=1}^{n_b} b_j + (2L + a_1). \quad (1)$$

$$F_{k+1}^{(2)} = n_b(L + X) + 2n_cL + 2 \sum_{j=1}^{n_c} c_j - \sum_{j=1}^{n_b} b_j + (2L + X - a_1). \quad (2)$$

$$F_{k+1}^{(3)} = n_c(5L + 2X) + \sum_{j=1}^{n_c} c_j. \quad (3)$$

**Case 1:**  $F_{k+1}^{(1)} \geq F_{k+1}^{(2)}$  and  $F_{k+1}^{(1)} \geq F_{k+1}^{(3)}$ .

From (1) and (2),

$$F_{k+1}^{(1)} \geq F_{k+1}^{(2)} \Rightarrow n_c \leq n_b. \quad (4)$$

From (1) and (3),

$$F_{k+1}^{(1)} \geq F_{k+1}^{(3)} \Rightarrow n_b \geq 2n_c - 1. \quad (5)$$

Since  $n_c \geq 1$ ,  $n_b \geq 2n_c - 1 \Rightarrow n_c \leq n_b$ . Therefore, we only need to focus on  $n_b \geq 2n_c - 1$ . Now, we have:

$$F_k^{(1)} = n_b(2L + 2X) + n_cL + \sum_{j=1}^{n_b} b_j, \quad (6)$$

$$F_k^{(2)} = n_b(L + X) + 2n_cL + 2 \sum_{j=1}^{n_c} c_j - \sum_{j=1}^{n_b} b_j. \quad (7)$$

Thus,

$$\begin{aligned} F_k^{(1)} - F_k^{(2)} &= (n_b - n_c)L + (n_b - 1)X + \left( X - 2 \sum_{j=1}^{n_c} c_j \right) + 2 \sum_{j=1}^{n_b} b_j \\ &> (n_b - n_c)L + (n_b - 1)X, \end{aligned} \quad (8)$$

due to  $L > nX > 2n \sum_{j=1}^n c_j = 2n \sum_{j=1}^n (a_j + b_j)$ . We have the following three cases:

Case 1(a):  $n_c = 1$

Since  $k \geq 3$  and  $n_c = 1$ , we must have  $n_b \geq 2$ . We can show that the first  $k$  orders in  $S_{OPT}$  must be of the pattern

$$S_{OPT} : b_1 b_2 c_1 (b)^{n_b-2}.$$

Any schedules that put  $c_1$  in positions other than the above can be converted into another one with smaller  $\sum C_j$ , contradicting the fact that  $S_{OPT}$  is optimal.

Let us consider the first  $(k + 1)$  orders in  $S_{OPT}$ . If  $n_b = 2$ , we have:

$$S_{OPT} : b_1 b_2 c_1 a_1.$$

Let us consider

$$S : a_1 b_2 c_1 b_1.$$

We have:

$$C_{b_1}(S_{OPT}) = \max\{2L + 2X + b_1, L + X - b_1, 0\} = 2L + 2X + b_1,$$

$$C_{b_2}(S_{OPT}) = \max\{4L + 4X + b_1 + b_2, 2L + 2X - b_1 - b_2, 0\} = 4L + 4X + b_1 + b_2,$$

$$C_{c_1}(S_{OPT}) = \max \left\{ \begin{array}{l} 5L + 4X + b_1 + b_2, \\ 4L + 2X + 2c_1 - b_1 - b_2, \\ 5L + 2X + c_1 \end{array} \right\} = 5L + 4X + b_1 + b_2.$$

and

$$C_{a_1}(S) = \max\{2L + a_1, 2L + X - a_1, 0\} = 2L + X - a_1,$$

$$C_{b_2}(S) = \max\{4L + 2X + a_1 + b_2, 3L + 2X - a_1 - b_2, 0\} = 4L + 2X + a_1 + b_2,$$

$$C_{c_1}(S) = \max \left\{ \begin{array}{l} 5L + 2X + a_1 + b_2, \\ 5L + 2X + 2c_1 - a_1 - b_2, \\ 5L + 2X + c_1 \end{array} \right\} < 5L + 2X + 2c_1 + a_1 + b_2.$$

Therefore,

$$\begin{aligned} \sum C_j(S) - \sum C_j(S_{OPT}) &= (C_{a_1}(S) + C_{b_2}(S) + C_{c_1}(S)) - \\ &\quad (C_{b_1}(S_{OPT}) + C_{b_2}(S_{OPT}) + C_{c_1}(S_{OPT})) \\ &< (11L + 5X + a_1 + 2b_2 + 2c_1) - (11L + 10X + 3b_1 + 2b_2) \\ &= -5X + a_1 + 2c_1 - 3b_1 < 0. \end{aligned}$$

Thus,  $S$  is better than  $S_{OPT}$ .

If  $n_b \geq 3$ , we have:

$$S_{OPT} : b_1 b_2 c_1 (b)^{n_b-3} b_{n_b} a_1.$$

We can show that

$$S : b_1 b_2 c_1 (b)^{n_b-3} a_1 b_{n_b}$$

is better than  $S_{OPT}$  by:

$$\begin{aligned} \sum C_j(S) - \sum C_j(S_{OPT}) &= C_{a_1}(S) - C_{b_{n_b}}(S_{OPT}) \\ &= \left( (n_b - 1)(2L + 2X) + \sum_{j=1}^{n_b-1} b_j + L + (2L + a_1) \right) - \left( n_b(2L + 2X) + \sum_{j=1}^{n_b} b_j + L \right) \\ &= -2X - b_{n_b} + a_1 < 0. \end{aligned}$$

Therefore, for Case 1(a), we can always find a schedule better than  $S_{OPT}$ .

Case 1(b):  $n_c = 2$

Since  $n_b \geq 2n_c - 1$ , we have  $n_b \geq 3$ . By the same argument as in Case 1(a), we can show that the first  $k$  orders in  $S_{OPT}$  must be scheduled in either

$$b_1 b_2 c_1 b_3 c_2 \quad (n_b = 3),$$

or

$$b_1 b_2 c_1 b_3 b_4 c_2 (b)^{n_b-4} \quad (n_b \geq 4).$$

In the former case, the schedule for the first  $(k + 1)$  orders is:

$$S_{OPT} : b_1 b_2 c_1 b_3 c_2 a_1.$$

We can show that

$$S : b_1 b_2 c_1 a_1 c_2 b_3$$

is better than  $S_{OPT}$  by:

$$\sum C_j(S) - \sum C_j(S_{OPT}) = (C_{a_1}(S) + C_{c_2}(S)) - (C_{b_3}(S_{OPT}) + C_{c_2}(S_{OPT})) = -2X + a_1 - b_3 < 0.$$

In the latter case, if  $n_b > 4$ , the schedule for the first  $(k + 1)$  orders is:

$$S_{OPT} : b_1 b_2 c_1 b_3 b_4 c_2 (b)^{n_b - 5} b_{n_b} a_1.$$

We can show that

$$S : b_1 b_2 c_1 b_3 b_4 c_2 (b)^{n_b - 5} a_1 b_{n_b}$$

is better than  $S_{OPT}$  by:

$$\begin{aligned} \sum C_j(S) - \sum C_j(S_{OPT}) &= C_{a_1}(S) - C_{b_{n_b}}(S_{OPT}) \\ &= \left( (n_b - 1)(2L + 2X) + \sum_{j=1}^{n_b - 1} b_j + 2L + (2L + a_1) \right) - \left( n_b(2L + 2X) + \sum_{j=1}^{n_b} b_j + 2L \right) \\ &= -2X + a_1 - b_{n_b} < 0, \end{aligned}$$

due to  $n_b > 4$ .

On the other hand, if  $n_b = 4$ , the schedule for the first  $(k + 1)$  orders is:

$$S_{OPT} : b_1 b_2 c_1 b_3 b_4 c_2 a_1.$$

We can show that

$$S : b_1 b_2 c_1 a_1 b_4 c_2 b_3$$

is better than  $S_{OPT}$  by:

$$\begin{aligned} \sum C_j(S) - \sum C_j(S_{OPT}) &= (C_{a_1}(S) + C_{b_4}(S) + C_{c_2}(S)) - (C_{b_3}(S_{OPT}) + C_{b_4}(S_{OPT}) + C_{c_2}(S_{OPT})) \\ &= -3(2X - a_1 + b_3) < 0. \end{aligned}$$

Therefore, for Case 1(b), we can always find a better schedule than  $S_{OPT}$ .

Case 1(c):  $n_c \geq 3$

Since  $n_b \geq 2n_c - 1$ , we have  $n_b \geq 5$ . From (8) and (5), we have:

$$\begin{aligned} F_k^{(1)} - F_k^{(2)} &> (n_b - n_c)L + (n_b - 1)X \\ &\geq (2n_c - 1 - n_c)L + 4X = (n_c - 1)L + 4X \geq 2L + 4X. \end{aligned} \quad (9)$$

If the  $k^{\text{th}}$  order is of type- $b$ , i.e., the schedule for the first  $(k + 1)$  orders is:

$$(b/c)^{k-1} b_{n_b} a_1,$$

then we have:

$$\begin{aligned}
& \left( F_{k-1}^{(1)} + (2L + 2X + b_{n_b}) \right) - \left( F_{k-1}^{(2)} + (L + X - b_{n_b}) \right) > 2L + 4X \\
& \Rightarrow \left( \left( F_{k-1}^{(1)} + 2L + 2X + b_{n_b} \right) + 2L + a_1 \right) - \left( \left( F_{k-1}^{(2)} + L + X - b_{n_b} \right) + 2L + X - a_1 \right) > \\
& \quad 2L + 4X + (2L + a_1) - (2L + X - a_1) \\
& \Rightarrow \left( F_{k-1}^{(1)} + 2L + a_1 \right) - \left( F_{k-1}^{(2)} + 2L + X - a_1 \right) > \\
& \quad 2L + 4X + (2L + a_1) - (2L + X - a_1) - (2L + 2X + b_{n_b}) + (L + X - b_{n_b}) \\
& \quad = L + 2X + 2a_1 - 2b_{n_b} > L + X. \tag{10}
\end{aligned}$$

This implies that, in  $(b/c)^{k-1}a_1b_{n_b}$ ,  $a_1$  will still finish on machine 1. However,  $(b/c)^{k-1}a_1b_{n_b}$  has a cost reduction by an amount of  $(2X + b_{n_b} - a_1)$  from  $(b/c)^{k-1}b_{n_b}a_1$ . Thus, this will result in a better schedule.

If the  $k^{th}$  order is of type- $c$ , let  $b_{n_b}$  be in the  $(k-l)^{th}$  ( $l \geq 1$ ) position, i.e., the pattern is:

$$(b/c)^{k-l-1}b_{n_b}(c)^l a_1.$$

We have:

$$\begin{aligned}
& \left( F_{k-l-1}^{(1)} + (2L + 2X + b_{n_b}) + lL \right) - \\
& \quad \left( F_{k-l-1}^{(2)} + (L + X - b_{n_b}) + 2lL + 2 \sum_{j=n_c-l+1}^{n_c} c_j \right) > 2L + 4X \\
& \Rightarrow \left( \left( F_{k-l-1}^{(1)} + 2L + 2X + b_{n_b} + lL \right) + 2L + a_1 \right) - \\
& \quad \left( \left( F_{k-l-1}^{(2)} + L + X - b_{n_b} + 2lL + 2 \sum_{j=n_c-l+1}^{n_c} c_j \right) + 2L + X - a_1 \right) > \\
& \quad 2L + 4X + (2L + a_1) - (2L + X - a_1) \\
& \Rightarrow \left( F_{k-l-1}^{(1)} + 2L + a_1 \right) - \left( F_{k-l-1}^{(2)} + 2L + X - a_1 \right) > \\
& \quad 2L + 4X + (2L + a_1) - (2L + X - a_1) - (2L + 2X + b_{n_b} + lL) + \\
& \quad \left( L + X - b_{n_b} + 2lL + 2 \sum_{j=n_c-l+1}^{n_c} c_j \right) \\
& \quad = (l+1)L + 2X + 2a_1 - 2b_{n_b} + 2 \sum_{j=n_c-l+1}^{n_c} c_j > (l+1)L + X. \tag{11}
\end{aligned}$$

This implies that if we swap  $a_1$  with  $b_{n_b}$ ,  $a_1$  will still finish on machine 1. But it results in a cost reduction by an amount of  $(2X + b_{n_b} - a_1)$ . In addition, for any type- $c$  order in position  $(k-l+r)$  ( $r = 1, 2, \dots, l$ ), we have:

$$\begin{aligned}
& \left( F_{k-l-1}^{(1)} + (2L + 2X + b_{n_b}) + rL + (l-r)L \right) - \\
& \quad \left( F_{k-l-1}^{(2)} + (L + X - b_{n_b}) + 2rL + 2 \sum_{j=n_c-l+1}^{n_c-l+r} c_j + 2(l-r)L + 2 \sum_{j=n_c-l+r+1}^{n_c} c_j \right) > 2L + 4X
\end{aligned}$$

$$\begin{aligned}
&\Rightarrow \left( F_{k-l-1}^{(1)} + (2L + 2X + b_{n_b}) + rL \right) - \left( F_{k-l-1}^{(2)} + (L + X - b_{n_b}) + 2rL + 2 \sum_{j=n_c-l+1}^{n_c-l+r} c_j \right) \\
&> 2L + 4X - (l-r)L + 2(l-r)L + 2 \sum_{j=n_c-l+r+1}^{n_c} c_j \\
&= 2L + 4X + (l-r)L + 2 \sum_{j=n_c-l+r+1}^{n_c} c_j \geq 2L + 4X. \tag{12}
\end{aligned}$$

Therefore, before the swap, for any position  $(k-l+r)$  ( $r = 1, 2, \dots, l$ ):

$$F_{k-l+r}^{(1)} - F_{k-l+r}^{(2)} > 2L + 4X.$$

This means that the type- $c$  order in position  $(k-l+r)$  finishes on machine 1 if  $F_{k-l+r}^{(1)} - F_{k-l+r}^{(3)} \geq 0$ ; otherwise it finishes on machine 3. Furthermore, since

$$\begin{aligned}
&\left( F_{k-l-1}^{(1)} + (2L + 2X + b_{n_b}) + rL + (l-r)L \right) - \\
&\quad \left( F_{k-l-1}^{(2)} + (L + X - b_{n_b}) + 2rL + 2 \sum_{j=n_c-l}^{n_c-l+r} c_j + 2(l-r)L + 2 \sum_{j=n_c-l+r+1}^{n_c} c_j \right) > 2L + 4X \\
&\Rightarrow \left( F_{k-l-1}^{(1)} + (2L + a_1) + rL \right) - \left( F_{k-l-1}^{(2)} + (2L + X - a_1) + 2rL + 2 \sum_{j=n_c-l}^{n_c-l+r} c_j \right) \\
&> 2L + 4X - (2L + 2X + b_{n_b}) + (2L + a_1) + (L + X - b_{n_b}) - (2L + X - a_1) \\
&\quad - (l-r)L + 2(l-r)L + 2 \sum_{j=n_c-l+r+1}^{n_c} c_j \\
&= L + (l-r)L + 2X + 2a_1 - 2b_{n_b} + 2 \sum_{j=n_c-l+r+1}^{n_c} c_j > L + X, \tag{13}
\end{aligned}$$

after the swap, for any position  $(k-l+r)$  ( $r = 1, 2, \dots, l$ ):

$$F_{k-l+r}^{(1)} - F_{k-l+r}^{(2)} > L + X.$$

This means that the type- $c$  order in position  $(k-l+r)$  still finishes on machine 1 if  $F_{k-l+r}^{(1)} - F_{k-l+r}^{(3)} \geq 0$ ; otherwise it finishes on machine 3. However, since  $F_{k-l+r}^{(1)}$  was reduced by an amount of  $(2X + b_{n_b} - a_1)$  after the swap, the finish time of the type- $c$  order in position  $(k-l+r)$  will either remain unchanged or be reduced.

The above argument shows that swapping  $a_1$  with  $b_{n_b}$  will result in a better schedule.

**Case 2:**  $F_{k+1}^{(2)} \geq F_{k+1}^{(1)}$  and  $F_{k+1}^{(2)} \geq F_{k+1}^{(3)}$ .

From (2) and (1),

$$F_{k+1}^{(1)} \leq F_{k+1}^{(2)} \Rightarrow n_b \leq n_c. \tag{14}$$

From (2) and (3),

$$F_{k+1}^{(3)} \leq F_{k+1}^{(2)} \Rightarrow n_c \leq \frac{n_b + 2}{3} \Rightarrow n_c \leq \frac{n_c + 2}{3} \Rightarrow n_c \leq 1, \tag{15}$$

due to (14). Thus,

$$(k = n_b + n_c) \wedge (n_b \leq n_c) \wedge (n_c \leq 1) \Rightarrow k \leq 2.$$

However, we have proved the case  $k \leq 2$  before. So this case needs not be considered any further.

**Case 3:**  $F_{k+1}^{(3)} \geq F_{k+1}^{(1)}$  and  $F_{k+1}^{(3)} \geq F_{k+1}^{(2)}$ .

In this case, the schedule must be of the pattern:

$$b_1(b/c)^l c_{n_c}(b)^{k-l-2} a_1.$$

Let us consider the following pattern:

$$b_1(b/c)^l (b)^{k-l-2} a_1 c_{n_c}.$$

It is clear that the finish time of each order in  $(b)^{k-l-2} a_1$  will be reduced by an amount of  $L$  on machine 1 and an amount of  $(2L + 2c_{n_c})$  on machine 2. The finish time of order  $c_{n_c}$  remains unchanged in its new position since it finishes on machine 3. Thus,  $b_1(b/c)^l (b)^{k-l-2} a_1 c_{n_c}$  is better than  $b_1(b/c)^l c_{n_c}(b)^{k-l-2} a_1$ .

Summarizing the above cases, the correctness of the observation follows. ■

*Observation 3.* In  $S_{OPT}$  the second order must be of type b.

**Proof.** Let  $(k + 1) > 2$  be the smallest position of a type- $b$  order in  $S_{OPT}$ . Following our convention, we denote this type- $b$  order as  $b_1$ . We will show that there is a schedule  $S$  which has  $b_1$  in a position less than  $(k + 1)$  but larger than 1, such that

$$\sum_{j=1}^{3n} C_j(S) < \sum_{j=1}^{3n} C_j(S_{OPT}).$$

In  $S_{OPT}$  the first order must be of type  $a$ , and the remaining  $(k - 1)$  orders are either of type  $a$  or type  $c$ . Let the number of type- $a$  and type- $c$  orders be  $n_a$  and  $n_b$ , respectively. Clearly,  $n_a + n_c = k$ .

If  $n_c = 0$ , it means that all of the first  $k$  orders are of type  $a$ . Thus, if  $k = 2$ , the schedule for the first  $(k + 1)$  orders is  $a_1 a_2 b_1$ . Consider the subsequence of  $S_{OPT}$  before  $c_1$ . We denote this subsequence as:

$$S_{OPT} : a_1 a_2 b_1 (a/b)^l c_1,$$

where  $l \geq 0$ . It can be shown that if  $S_{OPT}$  is changed to  $S$ :

$$S : a_1 b_1 c_1 (a/b)^l a_2,$$

then the  $\sum C_j$  is lower. Thus, we may assume that  $k \geq 3$ . For  $k \geq 3$ , the schedule for the first  $(k + 1)$  orders is:

$$S_{OPT} : a_1 (a)^{k-2} a_k b_1.$$

We can show that

$$S : a_1 (a)^{k-2} b_1 a_k$$

is better than  $S_{OPT}$ . Thus, we may assume that  $n_c \geq 1$ .

We may also assume that  $n_a \geq 2$ . Indeed, if  $n_a = 1$ , the schedule for the first  $(k+1)$  order is:

$$S_{OPT} : a_1 c_1 (c)^{k-2} b_1.$$

We can show that

$$S : a_1 b_1 c_1 (c)^{k-2}.$$

has smaller  $\sum C_j$  than  $S_{OPT}$ . Thus, we may assume that  $k = n_a + n_c \geq 3$ .

For  $k = 3$ , there are only two possible schedules for the first  $(k+1)$  orders:  $a_1 a_2 c_1 b_1$  and  $a_1 c_1 a_2 b_1$ . In the former case ( $a_1 a_2 c_1 b_1$ ), we can show that the schedule  $a_1 b_1 c_1 a_2$  is a better schedule. In the latter case ( $a_1 c_1 a_2 b_1$ ), we can show that the schedule  $a_1 c_1 a_2 b_1$  is a better schedule.

Thus, from now on, we only need to focus on the case  $k \geq 4$ .

First of all, for the first  $(k+1)$  orders, we have:

$$F_{k+1}^{(1)} = \left( 2n_a L + \sum_{j=1}^{n_a} a_j \right) + n_c L + (2L + 2X + b_1). \quad (16)$$

$$F_{k+1}^{(2)} = \left( 2n_a L + n_a X - \sum_{j=1}^{n_a} a_j \right) + \left( 2n_c L + 2 \sum_{j=1}^{n_c} c_j \right) + (L + X - b_1). \quad (17)$$

$$F_{k+1}^{(3)} = n_c (5L + 2X) + \sum_{j=1}^{n_c} c_j. \quad (18)$$

**Case 1:**  $F_{k+1}^{(1)} \geq F_{k+1}^{(2)}$  and  $F_{k+1}^{(1)} \geq F_{k+1}^{(3)}$ .

From (16) and (17),

$$F_{k+1}^{(1)} \geq F_{k+1}^{(2)} \Rightarrow (n_c = 0) \text{ or } (n_c = 1 \text{ and } n_a \leq 0). \quad (19)$$

Since we assume that  $n_c \geq 1$  and  $k \geq 4$ , this leads to a contradiction.

**Case 2:**  $F_{k+1}^{(2)} \geq F_{k+1}^{(1)}$  and  $F_{k+1}^{(2)} \geq F_{k+1}^{(3)}$ .

From (16) and (17),

$$F_{k+1}^{(1)} \leq F_{k+1}^{(2)} \Rightarrow (n_c = 1 \text{ and } n_a \geq 3) \text{ or } (n_c \geq 2). \quad (20)$$

It should be noted that, when  $n_c = 1$ , we let  $n_a \geq 3$ , since we only focus on  $k = n_a + n_c \geq 4$ .

From (17) and (18),

$$F_{k+1}^{(3)} \leq F_{k+1}^{(2)} \Rightarrow n_c \leq \frac{2n_a + 1}{3}. \quad (21)$$

Let us look at the order in the  $k^{\text{th}}$  position, we have:

$$F_k^{(1)} = \left( 2n_a L + \sum_{j=1}^{n_a} a_j \right) + n_c L. \quad (22)$$

$$F_k^{(2)} = \left( 2n_a L + n_a X - \sum_{j=1}^{n_a} a_j \right) + \left( 2n_c L + 2 \sum_{j=1}^{n_c} c_j \right). \quad (23)$$

Clearly, under the condition of (20),  $F_k^{(1)} < F_k^{(2)}$  always holds. We have the following two cases:

Case 2(a): The order in the  $k^{th}$  position is of type  $a$ .

In this case, the first  $(k+1)$  orders of the schedule is:

$$a_1(a/c)^{k-2} a_{n_a} b_1.$$

We can show that:

$$a_1(a/c)^{k-2} b_1 a_{n_a}.$$

is a better schedule.

Case 2(b): The order in the  $k^{th}$  position is of type  $c$ .

In this case, we prove by contradiction that the order in the  $(k-1)^{st}$  position must be of type  $a$ . Suppose that  $a_{n_a}$  is in the  $(k-l)^{th}$  ( $2 \leq l \leq n_c$ ) position, then the first  $k$  orders in  $S_{OPT}$  is:

$$S_{OPT} : a_1(a/c)^{k-l-2} a_{n_a} c_{n_c-l+1} c^{l-2} c_{n_c}.$$

We can show that  $S$ :

$$S : a_1(a/c)^{k-l-2} c_{n_c-l+1} a_{n_a} c^{l-2} c_{n_c}$$

is better than  $S_{OPT}$ .

Thus, the  $(k-1)^{st}$  order must be of type  $a$  if the  $k^{th}$  order is of type  $c$ . It follows that the first  $(k+1)$  orders are scheduled as:

$$S_{OPT} : a_1(a/c)^{k-3} a_{n_a} c_{n_c} b_1.$$

We can show that

$$S : a_1(a/c)^{k-3} b_1 c_{n_c} a_{n_a}$$

is better than  $S_{OPT}$

**Case 3**:  $F_{k+1}^{(3)} \geq F_{k+1}^{(1)}$  and  $F_{k+1}^{(3)} \geq F_{k+1}^{(2)}$ .

For this case, the first  $(k+1)$  orders in  $S_{OPT}$  must be of the pattern:

$$S_{OPT} : a_1(a/c)^l c_{n_c} (a)^{k-l-2} b_1.$$

We can show that the following pattern  $S$ :

$$S : a_1(a/c)^l (a)^{k-l-2} b_1 c_{n_c}.$$

has a smaller  $\sum C_j$  than  $S_{OPT}$ .

Summarizing the above cases, the correctness of the observation follows. ■

*Observation 4.* In  $S_{OPT}$  the third order must be of type  $c$ .

**Proof.** Let  $(k + 1) > 3$  be the smallest position of a type- $c$  order in  $S_{OPT}$ . Following our convention, we denote this type- $c$  order as  $c_1$ . We will show that there is a schedule  $S$  which has  $c_1$  in a position less than  $(k + 1)$  but larger than 2, such that

$$\sum_{j=1}^{3n} C_j(S) < \sum_{j=1}^{3n} C_j(S_{OPT}).$$

It is clear that the first  $k$  orders are either of type  $a$  or type  $b$ . Let the number of type- $a$  and type- $b$  orders be  $n_a$  and  $n_b$ , respectively. Clearly,

$$(k + 1) > 3 \Rightarrow k = n_a + n_b \geq 3.$$

Therefore,  $c_1$  must be finished on either machine 1 or machine 2.

For  $k = 3$ , there are two possible patterns for  $S_{OPT}$ :  $a_1b_1a_2c_1$  and  $a_1b_1b_2c_1$ . In the former case, we can show that  $a_1b_1c_1a_2$  is better than  $a_1b_1a_2c_1$ . In the latter case, we can show that  $a_1b_1c_1b_2$  is better than  $a_1b_1b_2c_1$ .

From now on, we can focus on  $k \geq 4$  only. There are two cases to consider:

**Case 1:**  $n_a \geq 2$ .

The pattern of the first  $(k + 1)$  orders in  $S_{OPT}$  is

$$S_{OPT} : a_1b_1(a/b)^{k-l-3}a_{a_n}b^l c_1.$$

We can show that  $S$ :

$$S : a_1b_1(a/b)^{k-l-3}c_1b^l a_{a_n}.$$

is better than  $S_{OPT}$ .

**Case 2:**  $n_a < 2$ .

$n_a < 2$  also means that  $n_a = 1$ , since the first order must be a type- $a$  order. Thus,

$$k = n_a + n_b \geq 4 \Rightarrow n_b \geq 3.$$

The pattern of the first  $(k + 1)$  orders in  $S_{OPT}$  is

$$S_{OPT} : a_1b_1(b)^{k-3}b_{b_n}c_1.$$

We can show that  $S$ :

$$S : a_1b_1(b)^{k-3}c_1b_{b_n}$$

is better than  $S_{OPT}$ .

Summarizing the above cases, the correctness of the observation follows. ■

We are now ready to show that the orders are scheduled in  $S_{OPT}$  in the order  $(abc)$ , repeated  $n$  times. This can be shown by induction on  $k$ , where  $k$  is the number of times that the  $(abc)$  pattern repeats. The previous observations have shown that the lemma is true for  $k = 1$ . The inductive step can be shown by the same argument as in the previous observations. This follows from the observations that (1) At the beginning, all three machines start at time

0; (2) After scheduling the  $(abc)$  pattern  $k$  times, the finish times of the three machines are  $5L + 2X + \alpha_1$ ,  $5L + 2X + \alpha_2$ , and  $5L + 2X + \alpha_3$ , respectively, where  $\alpha_i$  ( $1 \leq i \leq 3$ ) is a linear combination of the  $a_i$ 's,  $b_i$ 's and  $c_i$ 's. Since  $L$  and  $X$  are much larger than the  $a_i$ 's,  $b_i$ 's and  $c_i$ 's, we can view the finish times of the three machines as more or less equal. In other words, the  $a_i$ 's,  $b_i$ 's and  $c_i$ 's are inconsequential in the arguments.