TR-92-002

Corrections to "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment"[1]

Jui-Ching Cheng

November 15, 1991

[1]C. L. Liu and J. Layland, *J. ACM*, 20(1), 1973

Corrections to "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment"[1]

Jui-Ching Cheng

November 15, 1991

[1]C. L. Liu and J. Layland, *J. ACM*, 20(1), 1973

I. Introduction

In the original paper[1], there are two errors in the proof of theorem 4. The first lies in the following statements copied from the original paper.

Suppose that

$$C_1 = T_2 - T_1 - \Delta,\ \ \Delta > 0$$

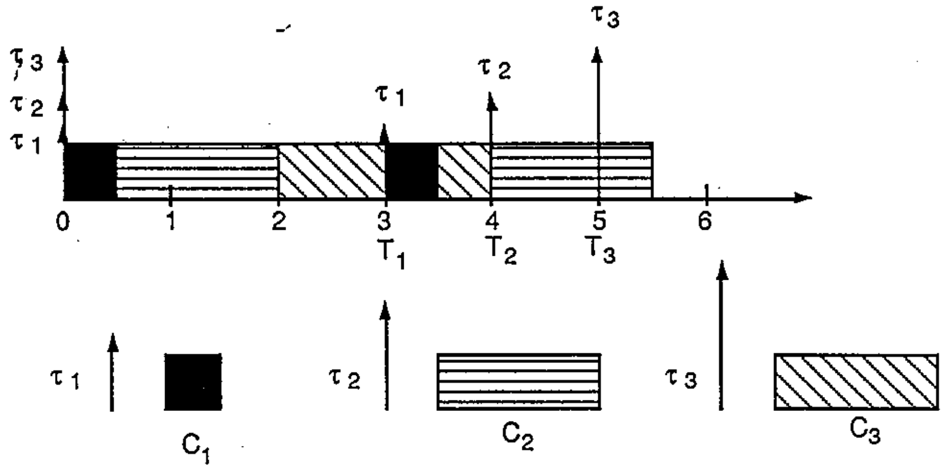
Let

$$\begin{aligned} C_1'' &= T_2 - T_1 \\ C_2'' &= C_2 - 2\Delta \\ C_3'' &= C_3 \\ &\ \ \vdots \\ C_m'' &= C_m \end{aligned} \qquad (1)$$

Again, C_1'', C_2'', C_3'', ..., C_m'' fully utilize the processor.

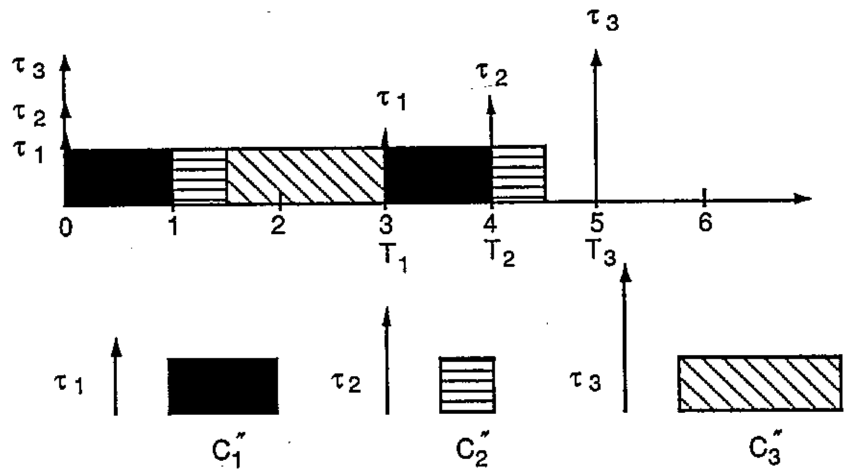
We find that equation (1) holds only when $C_2 > 2\Delta$. Furthermore, the resulting task set does not always fully utilize the processor. Consider the task set $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$, where

$$\begin{aligned} \tau_1 &: T_1 = 3, C_1 = 0.5 \\ \tau_2 &: T_2 = 4, C_2 = 1.5 \\ \tau_3 &: T_3 = 5, C_3 = 1.5\ . \end{aligned}$$



Obviously, this task set fully utilizes the processor. By changing C_1 , C_2 to C_1'' , C_2'' according to equation (1), we have a new task set $\mathcal{T}'' = \{\tau_1'', \tau_2'', \tau_3''\}$

$$\tau_1'' : T_1 = 3,\ C_1'' = 1$$
$$\tau_2'' : T_2 = 4,\ C_2'' = 0.5$$
$$\tau_3'' : T_3 = 5,\ C_3'' = 1.5\ .$$



Note that we can increase C_3'' by 0.5 while the resultant task set remains feasible. Thus, task set $\mathcal{T}''$ does not fully utilize the processor.

The second error lies in the proof of the utilization bound $U = m\,(2^{1/m} - 1)$. In the original paper, the authors use partial derivatives to find the minimum. We know that the point at which all partial derivatives equal zero is a local extreme, and not necessarily a global minimum.

Following is my proof of theorem 4.

II. Correction to Original Proof

Theorem 4. For a set of m tasks with fixed priority order, and the restriction that the ratio between any two request periods is less than 2, the least upper bound to the processor utilization factor is $U = m(2^{1/m} - 1)$.

Proof. Let $\mathcal{T}$ denote the task set consisting of $\tau_1, \tau_2, \tau_3, \ldots, \tau_m$, and $C_1, C_2, C_3, \ldots, C_m$ be the run-times of the task that fully utilize the processor. Assume that $T_m > T_{m-1} > \ldots > T_2 > T_1$. Let U denote the processor utilization factor.

First, we wish to show that when the following conditions are met, the utilization factor is minimum under the assumption that the periods are fixed and the run-times are allowed to change.

$$\begin{aligned} C_1 &= T_2 - T_1 \\ C_2 &= T_3 - T_2 \\ C_3 &= T_4 - T_3 \\ &\vdots \\ C_m &= T_{m+1} - T_m, \quad \text{where } T_{m+1} = 2T_1 \end{aligned}$$

The proof is divided by two stages.

Stage 1. We are going to show that if $C_j > T_{j+1} - T_j$ for some j, $1 \leq j \leq m-1$, then there exists $\mathcal{T}'$ that fully utilizes the processor with

$$C_i' \leq T_{i+1} - T_i, \text{ for all } 1 \leq i \leq m-1$$

and

$$U' < U,$$

where U' is the utilization factor of $\mathcal{T}'$ and U is the utilization factor of $\mathcal{T}$.

(i) Starting from τ_1, find the first task satisfying $C_i = T_{i+1} - T_i + \Delta$ and $\Delta > 0$. Let j-th task be this task. We define C_i' for $1 \leq i \leq m$ as follows

$$\begin{aligned} C_1' &= C_1 \\ C_2' &= C_2 \\ &\vdots \\ C_j' &= T_{j+1} - T_j \\ C_{j+1}' &= C_{j+1} + \Delta \\ &\vdots \\ C_m' &= C_m. \end{aligned}$$

Clearly, $C_1', C_2', C_3', \ldots, C_m'$ also fully utilize the processor. Let U' denote the corresponding utilization factor. It follows that

$$U - U' = \Delta/T_j - \Delta/T_{j+1} > 0.$$

That is

$$U' < U .$$

(ii) Starting from the j+1-th task, continue the search, and do the same transformation as (i) until the m-1-th task. Finally we get a new task set with run-time C_i' of each task i except τ_m' satisfying

$$C_i' \le (T_{i+1} - T_i) \text{ , for all } 1 \le i \le m\text{-}1$$

and still fully utilizes the processor. According to the proof of (i), the utilization factor of this new task set is also less than the original task set.

Stage 2. Let $\mathcal{T}'$ denote the resulting task set of stage 1. In this stage, we show that if $C_j' < T_{j+1} - T_j$ for some j, $1 \le j \le m\text{-}1$, then there exists $\mathcal{T}''$ that fully utilizes the processor with

$$C_i'' = T_{i+1} - T_i \text{ , for all } 1 \le i \le m \text{ , } T_{m+1} = 2\,T_1$$

and

$$U'' < U',$$

where U'' is the utilization factor of $\mathcal{T}''$ and U' is the utilization factor of $\mathcal{T}'$.

(i) Starting from τ_1', find the first task satisfying $C_j' = T_{j+1} - T_j - \Delta$ and $\Delta > 0$. Let j-th task be this task. We define C_i'' for $1 \le i \le m$ as follows

$$C_1'' = C_1'$$
$$C_2'' = C_2'$$
$$\vdots$$
$$C_j'' = T_{j+1} - T_j$$
$$\vdots$$
$$C_m'' = C_m' - 2\Delta .$$

For this set of task to be feasible and fully utilize the processor, the following two conditions must be met.

1. $C_m' \ge 2\Delta$.

2. Part of the execution time of τ_m' must be within $[T_j, T_m]$ with elapsing time $\ge \Delta$.

Since

$$C_i'' \le (T_{i+1} - T_i) \text{ , for } 1 \le i \le m\text{-}1, i \ne j$$

and

$$C_j'' = T_{j+1} - T_j - \Delta \text{ , } \Delta > 0,$$

it follows that

$$C_1' + C_2' + C_3' + \ldots + C_{m-1}' \le T_m - T_1 - \Delta < 2T_1 - T_1 - \Delta = T_1 - \Delta .$$

This means that the first jobs of $\tau_1', \tau_2', \tau_3', \ldots, \tau_{m-1}'$ completed before time $T_1 - \Delta$. Thus,we can assure that the run-time of τ_m' in the time zones $[0,T_1]$ and $[T_j,T_{j+1}]$ is greater than or equal to Δ, respectively (See remark 1) . This guarantees the above two conditions to be met while the resulting task set is still feasible and also fully utilizes the processor (See remark 2) .
Let U'' denote the corresponding utilization factor. Then we have

$$U' - U'' = -\Delta/T_j + 2\Delta/T_m > 0,$$

and equivalently

$$U'' < U' .$$

(ii) Starting from the j+1-th task, continue the search, and do the same transformation as (i) until the m-1-th task. Finally we get a new task set with run-time C_i^0 of each task i satisfying

$$C_i^0 = (T_{i+1} - T_i) \ \ , \text{for } 1 \le i \le m-1$$

and

$$C_m^0 = 2T_1 - T_m.$$

According to (i), this new task set also fully utilizes the processor and its utilization factor is also less than the original task set.

Thus, for a task set that fully utilizes the processor we have proved that if the periods are fixed and the run-times are allowed to change, the utilization factor is minimized when the following conditions are met

$$C_1^0 = T_2 - T_1$$

$$C_2^0 = T_3 - T_2$$

$$C_3^0 = T_4 - T_3$$

$$\vdots$$

$$C_m^0 = T_{m+1} - T_m, \quad \text{where } T_{m+1} = 2T_1 ,$$

and the utilization factor is

$$U^0 = \sum_{i=1}^{m} \frac{C_i}{T_i} = \sum_{i=1}^{m} \frac{T_{i+1}}{T_i} - m \qquad (2)$$

Now, we want to find the minimum value of U^0. Since the arithmetic mean is greater than or equal to the geometric mean of several positive real numbers, it follows that

$$U^0 = \sum_{i=1}^{m} \frac{C_i}{T_i} = \sum_{i=1}^{m} \frac{T_{i+1}}{T_i} - m \ge m \sqrt[m]{\prod_{i=1}^{m} \frac{T_{i+1}}{T_i}} - m .$$

Consequently,

$$U^0 \ge m\sqrt[m]{2} - m$$

and the equality holds when

$$\frac{T_{i+1}}{T_i} = \sqrt[m]{2} \text{ , for } i = 1 \text{ to } m \text{ .}$$

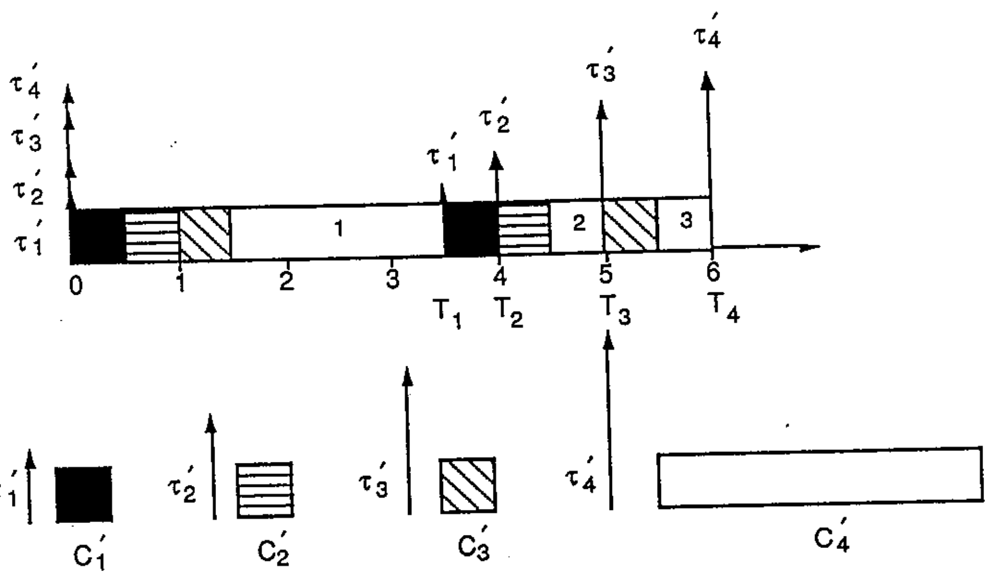
Thus, the proof is completed.

Acknowledgement. The author wish to thank Yuan Chen for his innovative suggestions and helps.

Remarks

1. In order to guarantee this condition, stage 1 must precede stage 2. That is, stage 2 may proceed only after stage 1 is completed. For example, consider the following intermediate task set obtained from stage 1

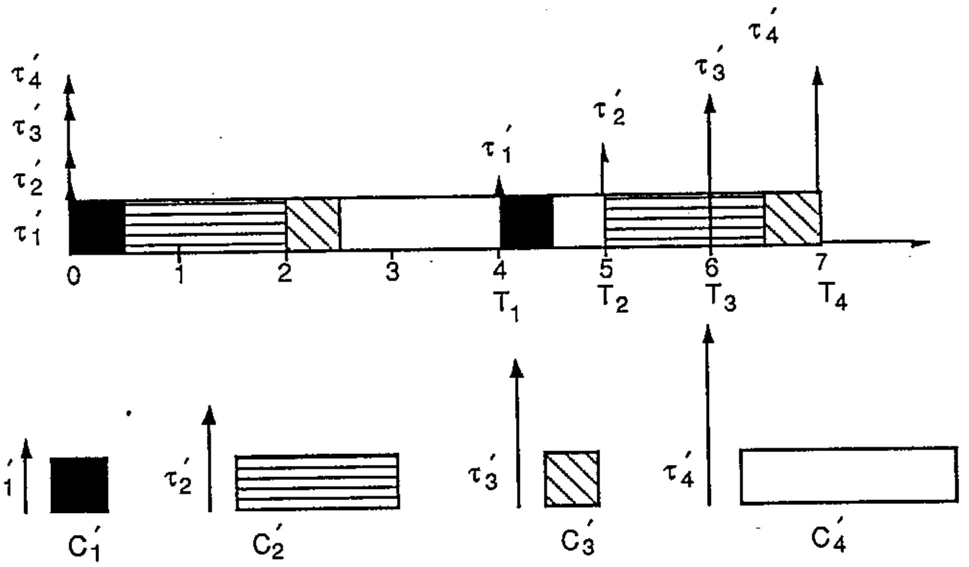
$$T_1 = 3.5,\ C_1' = 0.5 = T_2 - T_1$$
$$T_2 = 4,\quad C_2' = 0.5 = T_3 - T_2 - 0.5 = T_3 - T_2 - \Delta_2$$
$$T_3 = 5,\quad C_3' = 0.5 = T_4 - T_3 - 0.5 = T_4 - T_3 - \Delta_3$$
$$T_4 = 6,\quad C_4' = 3\ .$$



τ_4' runs in time zones 1, 2, and 3. Time zone 1 is greater than $\Delta_2 + \Delta_3$. Thus, there is sufficient time in time zones 1, 2, and 3 for C_2' and C_3' to increase. This assures that the task set produced by the manipulations of (i) still fully utilize the processor and is feasible.

2. If the operation of stage 2 starts before stage 1 is completed, the manipulations of (i) do not guarantee to produce a task set that fully utilize the processor and is feasible.Consider the following task set

$$\begin{aligned}
T_1 &= 4, & C_1' &= 0.5 = T_2 - T_1 - 0.5 = T_2 - T_1 - \Delta_1 \\
T_2 &= 5, & C_2' &= 1.5 = T_3 - T_2 + 0.5 = T_3 - T_2 + \Delta_2 \\
T_3 &= 6, & C_3' &= 0.5 = T_4 - T_3 - 0.5 = T_4 - T_3 - \Delta_3 \\
T_4 &= 7, & C_4' &= 2\,.
\end{aligned}$$



According to (i), C_3' must increase by 0.5 and C_4' decrease by 1. Thus, time zone [4.5,5] is idle and the resulting task set does not fully utilize the processor.

Reference

[1] C.L.Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", *J. ACM*, 20(1), 1973.