Bounds for Certain Multiprocessing Anomalies

By R. L. GRAHAM

(Manuscript received July 11, 1966)

It is known that in multiprocessing systems composed of many identical processing units operating in parallel, certain timing anomalies may occur; e.g., an increase in the number of processing units can cause an increase in the total length of time needed to process a fixed set of tasks. In this paper, precise bounds are derived for several anomalies of this type.

1. INTRODUCTION

In recent years there has been increased interest in the study of the potential advantages afforded by the use of a computer with many processors in parallel. While it is generally true that a set of tasks may be processed in less time by this type of multiprocessing, it has been pointed out that certain anomalies\(^1,2\) may occur, even though the processors are used in a very "natural" way (e.g., it can happen that increasing the number of processors can increase the time required to complete a given set of tasks).

It is the purpose of this paper to derive precise bounds on the extent to which these anomalies can affect the time required to process a set of tasks, given certain rather natural rules for the operation of the multiprocessing system.

1.1 Description of the System

Let us suppose that we are given \( n \) identical processing units \( P_i \), \( 1 \leq i \leq n \), and a set of tasks \( T = \{ T_1, \ldots, T_m \} \) to be processed by the \( P_i \). We are also given a partial-order* \(<\) on \( T \) and a function \( \mu: T \to [0, \infty) \). Once a processor \( P_i \) begins a task \( T_j \), it works without interruption on \( T_j \) until completion of that task, taking altogether \( \mu(T_j) \) units of time. It is also required that if \( T_i < T_j \) then \( T_j \) cannot

* See Ref. 2.
be started until \( T_i \) is completed. The \( P_i \) execute the \( T_j \) in the following way: We are given a linear ordering \( L: (T_{k_1}, \ldots, T_{k_m}) \) of \( T \) called a task list (or priority list). In general, at any time \( t \) a \( P_i \) completes a task, it immediately (and instantaneously) scans the list \( L \) (starting from the beginning) until it comes to the first task \( T_j \) which has not yet begun to be executed. If all the predecessors of \( T_j \) (i.e., those \( T_i < T_j \)) have been completed by time \( t \) then \( P_i \) begins working on \( T_j \). Otherwise \( P_i \) proceeds to the next task \( T_{j'} \), in \( L \) which has not yet begun to be executed, etc. If \( P_i \) proceeds through the entire list \( L \) without finding a task to execute then \( P_i \) becomes idle (we shall also say that \( P_i \) is working on an empty task). \( P_i \) remains idle until some other \( P_j \) completes a task at which time \( P_i \) (and of course \( P_j \)) immediately scans the list \( L \) as before for possible tasks to execute. If two processors \( P_i \) and \( P_j, i < j \), simultaneously attempt to begin the same task \( T_k \), it will be our convention to assign \( T_k \) to \( P_i \), the processor with the smaller index. The processors all start scanning \( L \) at time \( t = 0 \) and proceed in the above-mentioned fashion until some time \( \omega \), the least time for which all the tasks have been completed.

It will be helpful here to consider several examples. We shall indicate the partial-order \(< \) on \( T \) and the function \( \mu \) by a directed graph \( G(<,\mu) \). In \( G(<,\mu) \), the vertices will correspond to the \( T_j \) and a directed edge from \( T_i \) to \( T_j \) will indicate that \( T_i < T_j \). Each vertex of \( G(<,\mu) \) will actually be labelled with the symbol \( T_j/\mu(T_j) \), the \( \mu(T_j) \) indicating the time necessary to execute \( T_j \). The activity of each \( P_i \) is conveniently represented by a timing diagram \( \mathcal{G} \) (also known as a Gantt diagram; see Ref. 1). \( \mathcal{G} \) will consist of \( n \) horizontal half-lines (labelled by the \( P_i \)) in which each line is subdivided into segments* and labelled according to the state of the corresponding processor.

**Example 1:** \( n = 3 \), \( L: (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8) \)

\[
\begin{array}{c}
T_1/4 & T_2/3 \\
G(<,\mu): & \\
T_4/5 & T_5/1 & T_5/3 \\
T_6/2 & T_7/2 & T_8/3
\end{array}
\]

*We always consider the segments as being closed on the left and open on the right.
The symbol $\varphi_i$ indicates a processor is idle (i.e., working on the empty task $\varphi_i$) but not all the $T_j$ have been completed. The indexing of the $\varphi_i$ is arbitrary. Thus, for $\mathcal{G}$ we have $\omega = 9$.

**Example 2: $n = 4$, $L: (T_1, T_2, T_3, T_4, T_5)$**

Here, $\omega = 13$. Note that in this example, $\omega$ is independent of $L$. We should also point out here that we are using the convention that whenever any $T_j$ is completed, then all current empty tasks $\varphi_i$ are also terminated. Processors still idle are then given "new" empty tasks to complete (e.g., $P_4$ in Example 2).
Example 3: $n = 3$, $L: (T_1, T_2, T_3, T_4, T_5, T_6, T_7)$

\[
\begin{align*}
&T_1/1. \\
&T_2/1. \quad T_5/1 \\
&G(<, \mu): \\
&T_3/1. \quad T_6/1 \\
&T_4/1. \quad T_7/3
\end{align*}
\]

\[
\begin{array}{c|c|c|c}
  & T_1 & T_4 & T_7 \\
\hline
P_1 & 1 & 1 & 3 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
  & T_2 & T_5 \\
\hline
P_2 & 1 & 3 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
  & T_3 & T_6 \\
\hline
P_3 & 1 & 3 \\
\hline
\end{array}
\]

$\omega = 5$

Suppose we use a different list $L'$ given by $L': (T_1, T_2, T_7, T_3, T_4, T_5, T_6)$. We then have

\[
\begin{array}{c|c|c|c}
  & T_1 & T_3 & T_5 \\
\hline
P_1 & 1 & 1 & 1 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c|c}
  & T_2 & T_4 & T_6 \\
\hline
P_2 & 1 & 1 & 1 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c}
  & T_7 \\
\hline
P_3 & 3 \\
\hline
\end{array}
\]

$\omega' = 3$

Hence, by simply using a different list $L'$, we have shortened $\omega$ by nearly a factor of two. The significance of this and similar examples will be brought out in the next section.
We see that, in general, $\omega$ is a function of the task list $L$, the “time” function $\mu$, the partial-order $<$, and the number of processors $n$ (in addition to the rules under which the $P_i$ operate). In this note, we investigate the factor by which $\omega$ can increase if we simultaneously:

(i) Change* the task list $L$;
(ii) Decrease the function $\mu$;
(iii) Relax the partial-order $<$;
(iv) Change the number of processors from $n$ to $n'$.

While it might first be expected that (ii), (iii), or (iv) (with $n' > n$) would cause a decrease in $\omega$, easy counterexamples† show that is not always the case. In the next section we obtain an upper bound on the factor by which $\omega$ can increase because of (i), (ii), (iii), and (iv) (cf. Theorem, p. 1571). This bound is just the expression $1 + n - 1/n'$. We also show that this bound is the best possible in the sense that it cannot be replaced by any smaller function of $n$ and $n'$.

II. THE MAIN RESULTS

We begin this section by considering a special case of the general problem. We include this here in order to acquaint the reader with the basic ideas which will be used later. Suppose we are given a set of tasks $T = \{T_1, \cdots, T_m\}$ and a directed graph $G(<, \mu)$ giving a partial-order $<$ and a time function $\mu$ on $T$. We execute these tasks twice, each time using two identical processors $P_1$ and $P_2$. The first time the tasks are executed we use a task list $L$ while the second time the tasks are executed we use another task list $L'$. Suppose the corresponding finishing times are $\omega$ and $\omega'$. The question we consider now is this: How much can the ratio $\omega'/\omega$ vary? This is answered by the following

**Proposition:** $\frac{2}{3} \leq \frac{\omega'}{\omega} \leq \frac{3}{2}$.

**Proof:** By the symmetry of $\omega$ and $\omega'$ it suffices to show that $\omega'/\omega \leq \frac{3}{2}$. The basic idea we shall use is a simple one. Consider the timing diagram $\mathcal{G}$ obtained when the tasks are executed using the list $L$. We want to show that there is a chain‡ of tasks $T_{e_1} < T_{e_2} < \cdots < T_{e_r}$ which has the property that whenever a processor is idle (i.e., executing an empty task $\varphi_i$) then the other processor is executing one of the $T_{e_k}$.

* By “change” we mean “possibly change”, etc.
† As far as the author is aware, these facts were first pointed out by Richards.
‡ i.e., a linearly-ordered subset using the partial-order $<$. 
To define the \( T_{k} \) we proceed as follows. First, let \( T_{j_{1}} \) be defined to be the task which has the latest finishing time in \( \mathcal{G} \) (if there is more than one such task then we choose the task which is executed by the higher-indexed processor). Let \( \varphi_{i} \) be the empty task which has the latest finishing time of all those empty tasks which finish at a time not later than the starting time of \( T_{j_{1}} \). By the construction of \( \mathcal{G} \), there must be a task \( T_{u} \) which has the same finishing time as \( \varphi_{i} \). Define \( T_{j_{2}} \) to be \( T_{u} \).

In general, suppose we have defined \( T_{j_{k}} \) for some \( k \geq 2 \). To define \( T_{j_{k+1}} \), let \( \varphi_{i} \) be the empty task which has the latest finishing time of all those empty tasks \( \varphi_{i} \) which finish at a time not later than the starting time of \( T_{j_{k}} \). (If there are no such \( \varphi_{i} \) then we are done, i.e., \( T_{j_{k+1}} \) is not defined.) By hypothesis, there must be a task \( T_{u} \) which has the same finishing time as \( \varphi_{i_{k+1}} \) and which has a starting time not later than the starting time of \( \varphi_{i_{k+1}} \). Define \( T_{j_{k+1}} \) to be \( T_{u} \). We continue this algorithm for as long as possible, say, until we have defined \( T_{j_{1}}, \ldots, T_{j_{r}} \).

We first note that since no processor works on one empty task \( \varphi_{i} \), while the other processor works on more than one task, then at any time a processor is executing an empty task, \( \varphi_{i} \), the other processor is executing one of the \( T_{j_{k}} \). We next claim that \( T_{j_{k+1}} < T_{j_{k}} \) for \( 1 \leq k < r \). Suppose this is not the case. If \( t \) denotes the time at which a processor \( P_{i} \) started executing \( \varphi_{i_{k+1}} \) then by the hypothesis concerning the operation of the processors, \( P_{i} \) should not have been idle (i.e., working on \( \varphi_{i_{k+1}} \)) since at least one task, namely \( T_{j_{k}} \), was eligible to be executed at that time. Thus, the timing diagram \( \mathcal{G} \) is not valid and we have a contradiction. Hence, we must have \( T_{j_{k+1}} < T_{j_{k}} \) for \( 1 \leq k < r \). By defining \( T_{i_{k}} = T_{j_{k+1}} \) for \( 1 \leq k \leq r \), the first assertion is proved. It follows at once that if we let \( \mu(\varphi_{i}) \) denote the length of time a processor spends executing \( \varphi_{i} \), then

\[
\sum_{\varphi_{i} \in \mathcal{G}} \mu(\varphi_{i}) \leq \sum_{k=1}^{r} \mu(T_{j_{k}}).
\] (1)

The proof of the proposition now follows directly. Let \( T_{i_{1}} < T_{i_{2}} < \cdots < T_{i_{r}} \) be chosen (by the assertion just established) so that

\[
\sum_{\varphi_{i'} \in \mathcal{G}'} \mu(\varphi_{i'}) \leq \sum_{k=1}^{r} \mu(T_{i_{k}}),
\] (2)

where the \( \varphi_{i'} \) are taken from \( \mathcal{G}' \) (the timing diagram obtained when the list \( L' \) is used). Note that \( \omega' \) can be written as:

\[
\omega' = \frac{1}{2} \sum_{T_{k} \in \mathcal{T}} \mu(T_{k}) + \sum_{\varphi_{i'} \in \mathcal{G}'} \mu(\varphi_{i'}). 
\] (3)
From (2) and (3) we have
\[
\omega' \leq \frac{1}{2} \left( \sum_{T_k \in T} \mu(T_k) + \sum_{k=1}^{s} \mu(\phi'_i) \right).
\] (4)

Since the following inequalities hold:
\[
\omega \geq \frac{1}{2} \sum_{T_k \in T} \mu(T_k) \quad \text{(5)}
\]
\[
\omega \geq \sum_{k=1}^{s} \mu(T_{i_k}) \quad \text{(6)}
\]
(where (6) follows from the fact that \(T_{i_1} < T_{i_2} < \cdots < T_{i_s}\), then we have from (4), (5), and (6)
\[
\omega' \leq \frac{1}{2} (2\omega + \omega) = \frac{3\omega}{2}
\]
and the proposition follows.

The following example shows that the upper bound of \(\frac{3}{2}\) cannot be replaced by any smaller value.

Example 4: \(n = 2\), \(L: (T_1, T_3, T_2), L': (T_1, T_2, T_3)\)

\[
\begin{align*}
T_1/1 \\
G(<,\mu): \quad T_3/1 \\
T_2/2
\end{align*}
\]

\[
\begin{array}{c}
P_1 \\
\hline
T_1 & T_2 \\
\hline
1 & 1
\end{array}
\]

\[
\begin{array}{c}
P_2 \\
\hline
T_3 \\
\hline
2
\end{array}
\]

\[
\omega = 2
\]

\[
\begin{array}{c}
P_1 \\
\hline
T_1 & T_3 \\
\hline
1 & 2
\end{array}
\]

\[
\begin{array}{c}
P_2 \\
\hline
T_2 & \varphi_1 \\
\hline
1 & 2
\end{array}
\]

\[
\omega' = 3
\]

Therefore, \(\omega'/\omega = \frac{3}{2}\) and the upper bound of the proposition is achieved.
Before stating the main theorem we introduce some notation. Let \( T = \{ T_1, \cdots, T_m \} \) be a set of tasks. Let \( G = G(<, \mu) \) and \( G' = G'(<' , \mu') \) be two directed graphs for \( T \) with the partial-orders \(<, <' \) and the time functions \( \mu, \mu' \). We say that \( G \leq G' \) if:

(i) \( \mu' \leq \mu \), i.e., \( \mu'(T_j) \leq \mu(T_j) \) for all \( T_j \in T \).

(ii) \( <' \subseteq < \), i.e., \( T_i <' T_j \) implies \( T_i < T_j \) for all \( T_i, T_j \in T \).

Finally, suppose we execute the tasks twice, one time using the graph \( G \), a task list \( L \) and \( n \) processors, the other time using the graph \( G' \), a task list \( L' \) and \( n' \) processors. Let \( \omega \) and \( \omega' \) denote the respective finishing times. We then have the

Theorem: If \( G' \leq G \) then

\[
\frac{\omega'}{\omega} \leq 1 + \frac{n - 1}{n' - 1}.
\]

Proof: By a slight modification of the argument used in the proposition, it follows that if \( \varphi'_i, 1 \leq i \leq v \), denote the empty tasks of \( G' \) then there exists a chain of tasks \( T_{i_1} <' T_{i_2} <' \cdots <' T_{i_s} \) of tasks in \( T \) with the property that whenever a processor is idle then some other processor is executing one of the \( T_{i_k} \). From this we conclude

\[
\sum_{\varphi'_i \in G'} \mu'(\varphi'_i) \leq (n' - 1) \sum_{k=1}^{s} \mu'(T_{i_k}). \tag{7}
\]

As before we note that

\[
\begin{align*}
\omega' &= \frac{1}{n} \left( \sum_{T_j \in T} \mu'(T_j) + \sum_{\varphi'_i \in G'} \mu'(\varphi'_i) \right) \\
&\leq \frac{1}{n'} \left( \sum_{T_j \in T} \mu'(T_j) + (n' - 1) \sum_{k=1}^{s} \mu'(T_{i_k}) \right) \\
&\leq \frac{1}{n'} \left( \sum_{T_j \in T} \mu'(T_j) + (n' - 1) \sum_{k=1}^{s} \mu'(T_{i_k}) \right) \quad \tag{8}
\end{align*}
\]

where the inequality follows by (7). Since

\[
\omega \geq \frac{1}{n} \sum_{T_j \in T} \mu(T_j) \geq \frac{1}{n} \sum_{T_j \in T} \mu'(T_j) \tag{9}
\]

and

\[
\omega \geq \sum_{k=1}^{s} \mu(T_{i_k}) \geq \sum_{k=1}^{s} \mu'(T_{i_k}) \tag{10}
\]
then by (8), (9), and (10) we conclude
\[
\omega' \leq \frac{1}{n'} (n\omega + (n' - 1)\omega).
\]
Hence,
\[
\frac{\omega'}{\omega} \leq 1 + \frac{n - 1}{n'}
\]
and the theorem is proved.

To show that this bound is best possible, we give several examples, which show that the bound can be attained (to within $\varepsilon$) by varying any one of the four parameters $L$, $\mu$, $<$, or $n$.

**Example 5:** $L$ is varied.

\[n = n', \quad \mu = \mu', \quad < = <'.\]

\[L = (T_1, T_2, \ldots, T_{n-1}, T_{2n-1}, T_n, T_{n+1}, \ldots, T_{2n-2})\]

\[L' = (T_1, T_n, T_{n+1}, \ldots, T_{2n-2}, T_2, T_3, \ldots, T_{n-1}, T_{2n-1})\]

\[. T_1/1\]

\[. T_2/1\]

\[\vdots\]

\[. T_{n-1}/1\]

\[G(<,\mu): . T_n/n - 1\]

\[. T_{n+1}/n - 1\]

\[\vdots\]

\[. T_{2n-2}/n - 1\]

\[. T_{2n-1}/n\]
The diagram shows a sequence of nodes labeled $P_1, P_2, \ldots, P_n$ with transitions $T_1, T_2, \ldots, T_{2n-1}$.

For $P_1$ to $P_n$, the transitions are:

- $T_1$ with weight $1$ to $T_n$ with weight $n-1$.
- $T_2$ with weight $1$ to $T_{n+1}$ with weight $n-1$.
- $T_{n-1}$ with weight $1$ to $T_{2n-2}$ with weight $n-1$.
- $T_{2n-1}$ with weight $n$.

The transition matrix is given as:

\[
P_1 = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\
T_1 & T_2 & \cdots & T_{n-1} & T_{2n-1} & n & \cdots & n & n & n
\end{bmatrix}
\]

The transition weights are:

- $\omega = n$ for $P_1$ to $P_n$.
- $\omega' = 2n - 1$ for $P_1$ to $P_n$.

The labels $\varphi_1, \varphi_{n-2}, \varphi_{n-1}$ indicate the end states of the transitions.
Thus,

\[
\frac{\omega'}{\omega} = 2 - \frac{1}{n'},
\]

which is the value of \(1 + (n - 1)/n'\) when \(n = n'\).

**Example 6:** \(\mu\) is decreased.

\[n = n', \quad \mu \geq \mu', \quad < = <'\]

\[L = L': \quad (T_1, T_2, \ldots, T_{3n})\]

<table>
<thead>
<tr>
<th>(T_i)</th>
<th>(\mu(T_i))</th>
<th>(\mu'(T_i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_1)</td>
<td>2e</td>
<td>e</td>
</tr>
<tr>
<td>(T_2)</td>
<td>2e</td>
<td>e</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>(T_{n-1})</td>
<td>2e</td>
<td>e</td>
</tr>
<tr>
<td>(T_n)</td>
<td>2e</td>
<td>2e</td>
</tr>
<tr>
<td>(T_{n+1})</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(T_{n+2})</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>(T_{2n})</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(T_{2n+1})</td>
<td>(n - 1)</td>
<td>(n - 1)</td>
</tr>
<tr>
<td>(T_{2n+2})</td>
<td>(n - 1)</td>
<td>(n - 1)</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>(T_{3n})</td>
<td>(n - 1)</td>
<td>(n - 1)</td>
</tr>
</tbody>
</table>

(In \(G\),

\[T_i < T_j < T_{2n+1}\]

for \(1 \leq i \leq n\) and \(< j \leq 2n\).)
\[ \omega = n + 2e \]

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>( T_{i+1} )</td>
<td>( 2e )</td>
<td>( 1 )</td>
<td>( n - 1 )</td>
<td>( n - 1 )</td>
<td>( T_{2n+1} )</td>
<td>( T_{3n+1} )</td>
<td>( n - 1 )</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>( T_{i+2} )</td>
<td>( 2e )</td>
<td>( 1 )</td>
<td>( n - 1 )</td>
<td>( n - 1 )</td>
<td>( \ldots \ldots )</td>
<td>( T_{3n-1} )</td>
<td>( n - 1 )</td>
</tr>
<tr>
<td>( T_{n-1} )</td>
<td>( T_{3n-1} )</td>
<td>( 2e )</td>
<td>( 1 )</td>
<td>( n - 1 )</td>
<td>( n - 1 )</td>
<td>( T_{3n} )</td>
<td>( 2e )</td>
<td>( n - 1 )</td>
</tr>
</tbody>
</table>
\[
P_1: \begin{array}{cccc}
T_1 & T_{2n+2} & T_{2n} & T_{2n+1} \\
\varepsilon & n-1 & 1 & n-1
\end{array}
\]

\[
P_2: \begin{array}{cccc}
T_2 & T_{2n+3} & \varphi_1 & \varphi_n \\
\varepsilon & n-1 & 1 & n-1
\end{array}
\]

\[
P_{n-1}: \begin{array}{cccc}
T_{n-1} & T_{3n} & \phi_{n-2} & \phi_{2n-3} \\
\varepsilon & n-1 & 1 & n-1
\end{array}
\]

\[
P_n: \begin{array}{cccc}
T_n & T_{n+1} & T_{n+2} & \ldots & T_{2n-1} & \phi_{n-1} & \phi_{2n-2} \\
2\varepsilon & 1 & 1 & 1 & 1-\varepsilon & n-1
\end{array}
\]

\[G': \begin{array}{c}
\ldots \ldots \ldots \ldots \ldots \\
\varepsilon
\end{array}\]

\[
\omega' = 2n - 1 + \varepsilon
\]
Thus,

\[ \frac{\omega'}{\omega} = \frac{2n - 1 + \epsilon}{n + 2\epsilon} \]

which is arbitrarily close to \( 2 - (1/n) \) for \( \epsilon \) sufficiently small. We should note the interesting fact that \( \omega' \geq 2n - 1 + \epsilon \) for any list \( L' \) which may be used.

**Example 7:** < is relaxed.

\[ n = n', \quad \mu = \mu', \quad \preceq \subseteq \prec \prec', \]

\[ L = L': \quad (T_1, T_2, \cdots, T_{n(n-1)+2}) \]

\[ G(\prec', \mu): \quad T_1/\epsilon \]

\[ T_2/1 \quad T_3/1 \quad \vdots \quad T_{n(n-1)+1}/1 \]

\[ T_{n(n-1)+2}/n \]

\[ T_1/\epsilon \]

\[ T_2/1 \]

\[ G(\prec', \mu): \quad \vdots \]

\[ T_{n(n-1)+1}/1 \]

\[ T_{n(n-1)+2}/n \]
\[ P_1 \]
\[ \begin{array}{c}
T_1 & T_2 & T_{n+1} & \cdots & T_{n(n-2)+3} \\
1 & 1 & 1 & \frac{n}{1} & 1
\end{array} \]

\[ P_2 \]
\[ \begin{array}{c}
T_{n(n-1)+2} & \varphi_{n-1} \\
\frac{n}{\epsilon} & 1
\end{array} \]

\[ P_3 \]
\[ \begin{array}{c}
\varphi_1 & T_3 & T_{n+2} & \cdots & T_{n(n-2)+4} \\
1 & 1 & 1 & \frac{n}{1} & 1
\end{array} \]

\[ \mathcal{G} : \]
\[ \begin{array}{c}
\vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{array} \]

\[ \omega = n + \epsilon \]

\[ P_{n-1} \]
\[ \begin{array}{c}
\phi_{n-2} & T_{n-1} & T_{2n-2} & \cdots & T_{n(n-1)} \\
1 & 1 & 1 & \frac{n}{1} & 1
\end{array} \]

\[ P_n \]
\[ \begin{array}{c}
\phi_{n-2} & T_n & T_{2n-1} & \cdots & T_{n(n-1)+1} \\
1 & 1 & 1 & 1 & 1
\end{array} \]
Thus,
\[ \frac{\omega'}{\omega} = \frac{2n - 1}{n + \epsilon} \]

which is arbitrarily close to \( 2 - (1/n) \) for \( \epsilon \) sufficiently small.

**Example 8:** \( n \) is varied.

**Case 1:** \( n < n' \), \( \mu = \mu' \), \( < = <' \)

\[ L = L': (T_1, T_2, \cdots, T_{nn'-n'+n+2}) \]

\[ T_1/\epsilon \quad \cdots \quad T_{nn'-n'+n+2}/n' \]

\[ T_2/\epsilon \quad \cdots \quad T_{n+2}/1 \]

\[ G: \quad \vdots \quad \cdots \quad T_{n+3}/1 \]

\[ T_n/\epsilon \quad \cdots \quad \vdots \]

\[ T_{n+1}/\epsilon \quad \cdots \quad T_{nn'-n'+n+1}/1 \]

\[ P_1 \quad T_1 \quad T_{n+1} \quad T_{n+2} \quad \cdots \quad T_{nn'-n'+3} \]

\[ \vdots \quad \epsilon \quad \epsilon \quad 1 \quad 1 \]

\[ P_2 \quad T_2 \quad T_{nn'-n'+n+2} \quad \phi_{n-1} \]

\[ \epsilon \quad \cdots \quad n' \quad \epsilon \]

\[ G: \quad P_3 \quad T_3 \quad \phi_1 \quad T_{n+3} \quad \cdots \quad T_{nn'-n'+4} \]

\[ \epsilon \quad \epsilon \quad 1 \quad 1 \]

\[ \vdots \quad \cdots \quad \vdots \]

\[ P_n \quad T_n \quad \phi_{n-2} \quad T_{2n} \quad \cdots \quad T_{nn'-n'+n+1} \]

\[ \epsilon \quad \epsilon \quad 1 \quad 1 \]

\[ \omega = n' + 2\epsilon \]
Thus,

$$\frac{\omega'}{\omega} = \frac{n' + n - 1 + \varepsilon}{n' + 2\varepsilon}$$

which is arbitrarily close to $1 + (n - 1/n')$ for $\varepsilon$ sufficiently small.

*Case 2: $n > n'*. The construction in this case is similar to that of Case 1 and will not be presented.

We should note that in Example 8 we took $L = L'$. If it is of some consolation to a possibly battered intuition, it should be noted that if $n \leq n', \mu = \mu'$, and $\prec = \prec'$ then for any $L$ which is chosen, it is possible to choose a suitable $L'$ for which $\omega' \leq \omega$.

**III. CONCLUDING REMARKS**

It should be pointed out here that we have not considered models of the multiprocessor system in which the priority list $L$ is “dynamically formed” (as opposed to the fixed lists we have used thus far). For example, one seemingly quite reasonable way of doing this is as follows: At any time a processor is free, it immediately begins to execute the
“ready” task (i.e., one which has all its predecessors completed) which currently heads the longest chain of unexecuted tasks (including itself). Suppose by following this algorithm in choosing tasks, we have a finishing time of $\omega^*$. If we denote by $\omega_o$ the least possible finishing time (minimized over all lists), then we would like to assert something about the ratio $\omega^*/\omega_o$. It follows from what has been proved in this paper that $\omega^*/\omega_o \leq 2 - (1/n)$, (where $n$ is the number of processors) and we would hope that, in fact, we could show $\omega^*/\omega_o$ is considerably closer to 1 than this. Unfortunately, this is not possible since it can be shown that the best possible bound on this ratio is given by

$$\frac{\omega^*}{\omega_o} \leq 2 - \frac{2}{n + 1}.$$ 

It is interesting to note, however, that in the case in which the partial-order $<$ on the tasks is empty, then this bound can be improved* to

$$\frac{\omega^*}{\omega_o} \leq 3 - \frac{1}{3n},$$

which, again, is best possible.

In conclusion, one might ask just how "typical" the examples are for which $\omega^*/\omega_o$ is close to the upper bound $2 - (1/n)$. While very little work has been done on this aspect, empirical results (using computer simulation (see Ref. 1)) indicate that examples in which $\omega^*/\omega_o \geq 1.1$ are quite common.

IV. ACKNOWLEDGMENTS

I wish to acknowledge here the stimulating discussions I have had on this subject with S. Lin and with G. K. Manacher, who originally brought these questions to my attention.

REFERENCES

2. Oehsner, B. P., Controlling a Multiprocessor System, Bell Laboratories Record, February, 1966

* The proofs of this and the preceding result will appear in a later paper.