Scheduling Partially Ordered Tasks with Exponentially Distributed Times*

P. F. Reynolds
K. M. Chandy

TR-104 JULY 1979

* This work was supported by the National Science Foundation under Grant MCS74-13302

This report was reproduced with funds provided through the Hitachi Corp.

ABSTRACT

Consider a partially ordered set of tasks where each task has at most one immediate successor. The precedence graph for this task system is a forest. Assume that all task times are independent identically distributed exponential random variables. Then the B-schedule [2] minimizes the expected time to complete the task system on two identical processors. Schedules which are optimal when task times are deterministic are not necessarily optimal when task times are stochastic if the precedence graph is not a forest or if there are more than two processors.

It is necessary to study random task execution times since operating systems generally cannot precisely predict execution times. We present here the only known polynomially bounded algorithm for scheduling partially ordered tasks with non-deterministic task execution times.

KEYWORDS: Stochastic, precedence graph, scheduling, trees.

INTRODUCTION

There is substantial literature on the scheduling of partially ordered tasks with fixed task execution time where the goal is to minimize the time required to execute all tasks [2]. This paper investigates whether optimal deterministic schedules are also optimal in the case where task execution times are independent identically distributed (iid) exponential random variables, and where the objective is to minimize the expected time to process all tasks.

Consider a set of partially ordered tasks $T = \{T_1,\ldots,T_N\}$ and let G be its precedence graph. We shall use T to refer to the set of vertices in G, and we define the weight of task T_i in G to be the mean execution time for T_i. The weight of a directed path is defined to be the sum of the weights of all tasks along the path including the initial and final task. An exit task is defined as one with no successors. The level of an exit task is its weight. The level of a non-exit task is the weight of the maximum weight path from that task to an exit task. At any given time, an executable task is one all of whose predecessors have been completed. A Highest Levels First (HLF) schedule, is defined as: Whenever a processor becomes free, assign that executable task (if any) which is at the highest level of those executable tasks not yet assigned. A B-schedule [2] is an HLF schedule in which ties among executable highest level tasks are broken arbitrarily. An A-schedule is an HLF-schedule in which ties between highest level tasks are broken by a labeling scheme [2]. Hu [7] has shown that the B-schedule is an optimal non-preemptive schedule for forest precedence graphs (in which every task has at most one successor), and for an arbitrary number of processors, if all task execution times are deterministic and equal. Coffman and Graham [3] have shown that the A-schedule is an optimal non-preemptive schedule for arbitrary

task graphs given two processors and provided all task execution times are deterministic and equal. Muntz and Coffman [8,9] have shown that processor sharing all highest level tasks is an optimal preemptive schedule for tasks with arbitrary deterministic execution times and either two processors and an arbitrary precedence graph or an arbitrary number of processors and a forest precedence graph. Adam, Chandy and Dickson [1] have shown empirically that the B-schedule is near-optimal for arbitrary precedence graphs and stochastic task times.

In this paper we demonstrate first known results for precedence graphs in which task times are probabilistic. Section 1 contains results regarding the concept of flatness, a property which is used later to prove the optimality of HLF schedules for two processors and forest precedence graphs with independent, identically distributed (iid) exponential task times. The results of section 1 pertain to all forest precedence graphs in which task times are random variables with equal means.

In section 2 we demonstrate interesting properties of task graphs in which task times are iid exponential random variables. These results apply to general graphs and any number of processors. We know of no parallel to these results in deterministic scheduling theory.

Section 3 contains a demonstration of the exclusive optimality of the B-schedule (hereafter called HLF) for forest precedence graphs with iid exponential task times and two processors. Furthermore, it is demonstrated that preemption is not required in this case.

Sections 4 and 5 contain the results of our attempts to extend results in deterministic scheduling theory to probabilistic scheduling theory. In section 4 we demonstrate that Graham's bound [6] for the deterministic case not only applies to the probabilistic case, but also contains implications regarding a tendency for arbitrary schedules for probabilistic task graphs

to have a ratio significantly less than the stated bound. Section 5 is a collection of further attempted extensions which demonstrate that some but not all results in deterministic scheduling theory have a parallel in probabilistic scheduling theory.

1. Properties of Forests

1.1 Definitions

Consider a forest precedence graph (in which every task has at most one immediate successor), and where all task times are assumed to have equal means (without loss of generality we assume unity). We shall refer to such precedence graphs as Forest Precedence Graphs (FPGs). The key theorems in this section relate to the concept of flatness of FPGs. We now define this concept.

Let G and H be FPGs. Let N_i and M_i be the number of tasks at level i for G and H respectively, for i = 1,2,3,...,. Clearly N_i and M_i are non-negative integers. Let S(G,m) be the number of tasks at levels of m or higher in G. Then

$$S(G,m) = \sum_{i \geq m} N_i \qquad (1)$$

G is defined to be as flat as H, denoted by $G \sim H$, if and only if:

$$S(G,m) = S(H,m) \text{ for } m = 1,2,3,... \qquad (2)$$

Note, if $G \sim H$ then $H \sim G$.

G is defined to be as flat or flatter than H, denoted by $G \precsim H$, if and only if:

$$S(G,m) \leq S(H,m) \text{ for } m = 1,2,3,... \qquad (3)$$

Note, if $G \precsim H$ and $H \precsim G$ then $G \sim H$. Also if $G \precsim H$ and $H \precsim I$ then $G \precsim I$.

G is defined to be flatter than H, denoted by $G \prec H$, if and only if

(3) is true, and there exists some m, say m = n, such that

$$S(G,n) < S(H,n) \tag{4}$$

The FPG in Fig. 1 is as flat as the one in Fig. 3 and flatter than the one in Fig. 2. Note that the graphs of figs. 1 and 3 are not identical.

Let X be an HLF schedule on G which initially processes tasks x_1 and x_2 if there are at least two executable tasks in G and processes task x_1 if there is only one executable task in G. Let Y and Z be schedules on H; let Y be HLF and Z arbitrary. If H has at least two executable tasks let Y initially process tasks y_1 and y_2 and let Z initially process z_1 and z_2. If H has only one executable task let Y initially process y_1 and Z initially process z_1. Let the levels of x_i, y_i and z_i be j_i, k_i and ℓ_i respectively, i = 1,2. When j_2,k_2 and ℓ_2 exist, assume without loss of generality that

$$j_1 \geq j_2,\ k_1 \geq k_2 \text{ and } \ell_1 \geq \ell_2 \tag{5}$$

Let $G - x_i$ be the subgraph of G obtained by deleting x_i. We define $H - y_i$ and $H - z_i$ similarly.

If $N_i \leq 1$ all i, then there exists at most one executable task in G. Since X is HLF,

$$j_1 = \max \{i \mid N_i \geq 1\} \tag{6}$$

If there exists some i such that $N_i \geq 2$, then

$$j_2 = \max \{i \mid N_i \geq 2\} \tag{7}$$

Similarly, since Y is HLF

$$k_1 = \max \{i \mid M_i \geq 1\} \tag{8}$$

If there exists some i such that $M_i \geq 2$, then

$$k_2 = \max \{i \mid M_i \geq 2\} \tag{9}$$

Note that (3), (6) and (8) imply that if $G \lesssim H$ then

$$k_1 \geq j_1 \tag{10}$$

Furthermore, it is obvious from (8) and (9) that

$$k_1 \geq \ell_1 \text{ and } k_2 \geq \ell_2 \tag{11}$$

1.2 Theory

Lemma 1

Let H have two or more executable tasks. Then

$$H - y_1 \preceq H - z_1 \tag{12}$$

and $$H - y_2 \preceq h - z_2 \tag{13}$$

Furthermore, if

$$k_1 > \ell_1 \text{ then } H - y_1 \prec H - z_1 \tag{14}$$

and if

$$k_2 > \ell_2 \text{ then } H - y_2 \prec H - z_2 \tag{15}$$

Proof: $$S(H-y_1,m) = \begin{cases} S(H,m) - 1 \text{ for } m \leq k_1 & (16) \\ S(H,m) = 0 \text{ for } m > k_1 & (17) \end{cases}$$

and

$$S(H - z_1,m) = \begin{cases} S(H,m) - 1 \text{ for } m \leq \ell_1 & (18) \\ S(H,m) \text{ for } m > \ell_1 & (19) \end{cases}$$

From eqns (11), (16) - (19) we get

$$S(H - y_1,m) \leq S(H - z_1,m) \text{ all } m \tag{20}$$

Eqn (20) is equivalent to (12). The proof for (13) is similar.

If $k_1 > \ell_1$ then

$$S(H - y_1,k_1) = S(H,k_1) - 1 < S(h,k_1) = S(H - z_1,k_1) \tag{21}$$

(21) and (12) imply (14). By a similar argument (15) holds when $k_2 > \ell_2$.

Lemma 2

Let G and H have two or more executable tasks. If $G \preceq H$ then

$$G - x_1 \preceq H - y_1 \tag{22}$$

and $$G - x_2 \preceq H - y_2 \tag{23}$$

Furthermore, if $G \prec H$ then either

$$G - x_1 \prec H - y_1 \tag{24}$$

or $$G - x_2 \prec H - y_2 \tag{25}$$

or both (24) and (25) are true.

Proof: We first prove (23); the proof for (22) is similar. Consider two cases: (1) $j_1 = j_2$ and (2) $j_1 > j_2$

Case 1, $j_1 = j_2$

$$S(G-x_2, m) = \begin{cases} S(G,m) - 1 \text{ for } m \leq j_1 = j_2 & (26) \\ S(G,m) = 0 \text{ for } m > j_2 \text{ (since } j_1 = j_2) & (27) \end{cases}$$

Also,

$$S(H-y_2,m) = \begin{cases} S(H,m) - 1 \text{ for } m \leq k_2 & (28) \\ S(H,m) \quad \text{for } m > k_2 & (29) \end{cases}$$

Recall from (3) that if $G \preceq H$ then,

$$S(G,m) \leq S(H,m) \text{ all } m \quad (30)$$

Also note that the sum of tasks in any task graph is non-negative. In particular,

$$S(H-y_2,m) \geq 0 \text{ all } m \quad (31)$$

We now consider two subcases: (a) $j_2 \leq k_2$ and (b) $j_2 > k_2$

Subcase a: $j_2 \leq k_2$

From (26), (28) and (30),

$$S(G-x_2,m) = S(G,m) - 1 \leq S(h,m) - 1 = S(H-y_2,m) \text{ for } m \leq j_2 \leq k_2 \quad (32)$$

From (27), and (31),

$$S(G-x_2,m) = 0 \leq S(H-y_2,m) \text{ for } j_2 < m \quad (33)$$

(32) and (33) imply

$$S(G-x_2,m) \leq S(H-y_2,m) \text{ for all } m \text{ for subcase (a)} \quad (34)$$

This proves (23) for subcase (a)

Subcase b: $j_2 > k_2$

From (26), (28) and (30),

$$S(G-x_2,m) = S(G,m) - 1 \leq S(H,m) - 1 = S(H-y_2,m) \text{ for } m \leq k_2 < j_2 \quad (35)$$

From (26), (29) and (30),

$$S(G-x_2,m) = S(G,m) - 1 \leq S(H,m) = S(H-y_2,m) \quad \text{For } k_2 < m \leq j_2 \quad (36)$$

And from (27) and (31),

$$S(G-x_2,m) = 0 \leq S(H-y_2,m) \text{ for } k_2 < j_2 < m \quad (37)$$

(35), (36) and (37) imply (23) for $j_2 > k_2$

This completes the proof of case 1.

Case 2 $j_1 > j_2$

At each level i, where $j_1 \geq i > j_2$, there is exactly one task in G, see (6) and (7).

Hence

$$S(G - x_2,m) = \begin{cases} S(G,m) - 1 \text{ for } m \leq j_2 & (38) \\ S(G,m) = 1 + j_1 - m \text{ for } j_2 < m \leq j_1 & (39) \\ S(G,m) = 0 \text{ for } m > j_1 & (40) \end{cases}$$

Using the same arguments as in case 1, we derive from (32), (35) and (36) that

$$S(G - x_2,m) \leq S(H - y_2,m) \text{ for } m \leq j_2 \quad (41)$$

From (31) and (40),

$$S(G - x_2,m) = 0 \leq S(H - y_2,m) \text{ for } j_1 < m \quad (42)$$

From (8) and (9) there must be at least one task at each level i, $i \leq k_1$ in $H - y_2$. Hence,

$$S(H - y_2,m) \geq 1 + k_1 - m \text{ for } m \leq k_1 \quad (43)$$

From (10), (39) and (43)

$$S(G - x_2,m) = 1 + j_1 - m \leq 1 + k_1 - m \leq S(H - y_2,m) \text{ for } j_2 < m \leq j_1 \quad (44)$$

(40), (42) and (44) imply (23).

We now prove at least one of (24) or (25) is true. If $G \prec H$, then there exists some m, say m = n, such that:

$$S(G,n) < S(H,n) \quad (45)$$

Clearly $n \leq k_1$ since $S(H,n) = 0$ for $n > k_1$.

Consider two cases: (1) $n \leq j_1$ (2) $j_1 < n \leq k_1$

Case 1. Since $n \leq j_1$, $S(G - x_1,n) = S(G,n) - 1$. Similarly, $S(H - y_1,n) = S(H,n) - 1$. Hence from (45),

$$S(G - x_1,n) = S(G,n) - 1 < S(H,n) - 1 = S(H - y_1,n) \quad (46)$$

(46) and (22) imply (24)

Case 2. Since $n > j_1 \geq j_2$

$$S(G - x_2, n) = S(G, n) = 0 \tag{47}$$

From (43)

$$S(H - y_2, n) > 1 + k_1 - n > 0 \tag{48}$$

(47) and (48) imply

$$S(G - x_2, n) < S(H - y_2, n) \tag{49}$$

(49) and (23) imply (25)

Lemma 3

If $G \sim H$ then $G - x_1 \sim H - y_1$, $i = 1,2$

Proof: If $G \sim H$ then $G \preceq H$ and $H \preceq G$, and hence $G - x_i \preceq H - y_i$, $i = 1,2$ and $H - y_i \preceq G - x_i$, $i = 1,2$ which implies

$$G - x_i \sim H - y_i, \quad i = 1,2.$$

Theorem 1

Let G and H have two or more executable tasks. If $G \preceq H$ then

$$G - x_1 \preceq H - z_1 \tag{50}$$

and $$G - x_2 \preceq H - z_2 \tag{51}$$

Furthermore, if $G \prec H$ then either

$$G - x_1 \prec H - z_1 \tag{52}$$

or $$G - x_2 \prec H - z_2 \tag{53}$$

or both (52) and (53) are true.

Proof: (50) follows from (12) and (22), while (51) follows from (13) and (23). (12), (13) and either (24) or (25) or both implies either (52) or (53) or both.

2. Properties of Partially Ordered Tasks in which Task Times are Independent Exponential Random Variables

In this section we shall consider arbitrary graphs (not necessarily forests) in which task execution times are independent exponential random variables. We will allow an arbitrary number of processors, and not all processors need have the same processing speed. In this section we only consider preemptive schedules in which there is no delay associated with preemption.

We prove several lemmas which have no counterpart in deterministic scheduling theory. It is surprising that stronger results can be obtained in the probabilistic case than in the deterministic case.

The time required to complete a set of n tasks is specified by the partial ordering over the tasks (usually specified by a graph G), the distribution $F(t_1,..,t_n)$ where t_i is the execution time of the ith task, and the processing schedule. At any instant of time, while processing is proceeding, some tasks may have been completed, others may be partially processed and processing may not have begun on others. The time required to complete all tasks will depend on the partial ordering over the remaining tasks and on the distribution of <u>remaining</u> times to complete execution of these tasks. The distribution of remaining task execution times depends on the distribution of the times required to process the unprocessed tasks and, in general, on the lengths of time for which the partially processed tasks have been processed. We show in the following lemma that if task times are independent, exponential random variables, the distribution of remaining execution times is independent of the amount of processing received by partially processed tasks. Hence, to compute the time required by a schedule we may restrict attention to the partial ordering (or graph) of remaining tasks.

Lemma 4

The remaining times to complete execution of partially processed tasks are independent of the amount of processing received so far by these tasks when task execution times are independent and exponential.

Proof: Assume at some instant that tasks $j(1),\ldots,j(k)$ are incomplete (i.e.either partially processed or unprocessed), and that task $j(i)$ has received $x(j(i))$ units of processing time at this instant ($x(j(i))=0$ if $j(i)$ is unprocessed). Let $F(t_{j(1)},\ldots,t_{j(k)} \mid x_{j(1)},\ldots,x_{j(k)})$ be the distribution of remaining execution times for tasks $j(i),\ldots,j(k)$ given $x(j(i))$ units of processing time recevied by task $j(i)$, $i = 1,\ldots,k$. Since task execution times are independent

$$F(t_{j(1)},\ldots,t_{j(k)} \mid x_{j(1)}, x_{j(k)}) = \prod_{i=1}^{k} F(t_{j(i)} \mid x_{j(i)})$$

It is well known [5] that

$$F(t_{j(i)} \mid x_{j(i)}) = F(t_{j(i)})$$

where $F(t_{j(i)})$ is the unconditioned distribution, if and only if $t_{j(i)}$ is an exponential random variable.

Hence:

$$F(t_{j(1)},\ldots,t_{j(k)} \mid x_{j(1)},\ldots, x_{j(k)}) = \prod_{i=1}^{k} F(t_{j(i)})$$

for all values of $x_{j(1)},\ldots,x_{j(k)}$.

A set of partially processed tasks in which task execution times are independent exponential random variables is completely specified by the partial ordering and the mean execution time of each task (since the mean value uniquely determines the distribution for exponential random variables). Assuming that a graph G specifies the partial ordering of partially processed tasks and that each vertex is labeled with the mean execution time of the corresponding task we see that G completely specifies the set of tasks, whether they are partially processed or unprocessed. We shall now restrict our attention to the case where remaining execution times

are independent exponential random variables and where any set of tasks is specified by a graph.

Let G be a graph specifying a set of tasks and let S be a schedule. Define T(G,S) as the mean time required to complete (all tasks in) G using schedule S. We now show that we may restrict attention to schedules in which a task may be preempted only at those instants at which another task is completed. (Note that there does not exist a similar property in deterministic scheduling). The intuition behind the restriction is simple: if the optimal policy is to make some assignment of processors to tasks initially for a graph G, then an optimal policy is to continue with the same assignment until G changes, because the optimal policy depends only on the graph and is independent of the amount of processing received by partially processed graphs. We now present a formal argument.

Let S be an optimum schedule which initially makes some assignment of processors to a set of tasks $\bar{a}$ in G and if none of the tasks finishes in some arbitrary time τ it preempts one or more partially processed tasks and switches to another schedule S'. Let S'' be a schedule which is derived from S by setting τ to infinity, i.e. the initial assignment is continued until at least one of the tasks in $\bar{a}$ is completed (for purposes of symmetry we may think of S' as derived from S by setting τ to zero). Note that processors may have different speeds and G is any arbitrary graph.

Lemma 5 $T(G,S) = p \cdot T(G,s') + (1-p) \cdot T(G,S'')$ where p is the probability that none of the tasks in $\bar{a}$ finishes in time τ.

Proof: Let $\bar{a} = \{a_1,\ldots,a_k\}$ and let the mean time required to complete a_i given the initial assignment of processors in S be $1/\lambda_i$. The probability that task a_i will finish in the incremental interval $(t,t + \Delta t)$ where $t < \tau$,

given schedule S, before any of the other tasks in $\bar{a}$ finishes is $\lambda_i e^{-\lambda t}$. where $\lambda = \sum_{i=1}^{k} \lambda_i$ (see [5] or [4]). If a_i finishes in the interval $(t, t + \Delta t)$, before any other task in $\bar{a}$, then the total execution time from time 0(zero) will be: $t + T(G-a_i,S)$. If none of the tasks finishes in time τ, the total execution time will be $\tau + T(G,S')$. The probability that none of the tasks finshes in τ is $e^{-\lambda\tau}$. Hence

$$T(G,S) = \sum_{i=1}^{k} \int_{0}^{\tau} (t + T(G-a_i,S))\ \lambda_i\ e^{-\lambda t}\ dt$$

$$+ e^{-\lambda\tau}\ (\tau + T(G,S'))$$

Setting $\tau = \infty$, we get

$$T(G,S'') = \sum_{i=1}^{k} \int_{0}^{\infty} (t + T(G-a_i,S))\ \lambda_i e^{-\lambda t}\ dt$$

After some symbol manipulation we get

$$T(G,S) = e^{-\lambda\tau}.\ T(G,S') + (1-e^{-\lambda\tau})\ T(G,S'') \qquad (54)$$

Lemma 6 Either S' or S'' is also an optimum schedule.

Proof: Assume S' is not an optimum schedule, i.e. $T(G,S') > T(G,S)$.

Then from (54), $T(G,S) > T(G,S'')$, and hence S'' is an optimum schedule.

Theorem 2 There exists an optimum schedule in which a task is preempted only at those instants when another task is completed.

Proof: Given any optimal schedule S in which a set of tasks is preempted after arbitrary time τ, if none of them completes we can create another optimal schedule in which $\tau = 0$ or $\tau = \infty$. In this manner we can create an optimal schedule in which a task is preempted only when another task is completed.

In the following lemma we prove that a subgraph of a graph G has an expected time to complete all tasks which is strictly less than the

expected time to complete all tasks in G. We note that there does not exist an equivalent result in deterministic scheduling theory.

Lemma 7 Let G be any task graph in which task times are independent exponential random variables and let b be some executable task in G. Let T(G,Opt) be the mean time required to complete G using an optimum schedule. Then,

$$T(G-b,Opt) < T(G, Opt) \tag{55}$$

Proof: By induction: The lemma is trivially true if G consists of a single task, b, since $T(G-b,Opt) = 0 < T(b,Opt)$. Assume the lemma to be true for all graphs G consisting of n or fewer tasks. We shall prove the lemma true for G having n + 1 tasks.

Let the optimum schedule for G initially assign processors to a set of tasks $\bar{a} = \{a_1,\ldots,a_k\}$. Consider two cases: (1) b is not in $\bar{a}$ and (2) b is in $\bar{a}$.

Case 1: Consider a schedule S for G-b which makes the same initial assignment of processors to tasks as the optimum schedule for G, and after one of the tasks in $\bar{a}$ finishes it follows the optimal schedule. Let the mean execution time for task a_i given this assignment be $1/\lambda_i$. The mean time until the first of the tasks in $\bar{a}$ finishes is $1/\lambda$ where $\lambda = \sum_{i=1}^{k} \lambda_i$ (see [5]) The probability that task a_i is the first task in $\bar{a}$ to finish is λ_i / λ (see [5]). Hence:

$$T(G,Opt) = \frac{1}{\lambda} + \sum_{i=1}^{k} \frac{\lambda}{\lambda}i \; . \; T(G-a_i,Opt) \tag{56}$$

and

$$T(G-b,S) = \frac{1}{\lambda} + \sum_{i=1}^{k} \frac{\lambda}{\lambda}i \; . \; T(G-a_i-b,Opt)$$

Since

$$T(G - a_i - b,Opt) < T(G-a_i,Opt) \text{ all } i$$

by the induction assumption, it follows that

$$T(G-b,S) < T(G,Opt)$$

By definition $T(G-b,Opt) \leq T(g-b,S)$; hence (55) follows.

Case 2: Assume without loss of generality that $b = a_1$. Consider a schedule S for $G - a_1$ which initially makes the same assignment of processors to tasks except that the processor assigned to a_1 in the optimum schedule for G is not assigned to any task in S. Let $\lambda' = \sum_{i=2}^{k} \lambda_i$

$T(G,Opt)$ is given by (56), and

$$T(G-a_1,S) = \frac{1}{\lambda'} + \sum_{i=2}^{k} \frac{\lambda i}{\lambda'} \cdot T(G-a_i-a_1,Opt) \qquad (57)$$

Multiplying (57) by $\lambda' \cdot \lambda_1$, recalling that $T(G-a_1,S) \geq T(G-a_1,Opt)$ and adding $\lambda' + \sum_{i=2}^{k} \lambda' \cdot \lambda_i \cdot T(G-a_i,Opt)$ to both sides we get

$$\lambda' + \sum_{i=1}^{k} \lambda' \cdot \lambda_i \cdot T(G-a_i,Opt) = \lambda + \sum_{i=2}^{k} \lambda_i \quad (\lambda' T(G-a_i, Opt) + \lambda_1 T(G-a_i-a_1,Opt)). \qquad (58)$$

$$T(G-a_i,Opt) > T(G-a_i-a_1,Opt) \text{ for } i=2,\ldots,k$$

by the induction assumption. Substituting this in (58), we get

$$\lambda' + \sum_{i=1}^{k} \lambda' \cdot \lambda_i \cdot T(G-a_i,Opt) > \lambda + \sum_{i=2}^{k} \lambda_i \cdot \lambda \cdot T(G-a_i-a_1,Opt)$$

Dividing by $\lambda'\lambda$ and using (56) and (57) gives

$$T(G,Opt) > T(G-a_1,S) \qquad (59)$$

By definition $T(G-b,Opt) = T(G-a_1,Opt) \leq T(G-a_1,S)$ (60)

Equation (55) follows from (59) and (60).

Lemma 8 Let A be a non-empty subset of tasks in g such that a task in A either has no predecessors or has all its predecessors in A, Then,

$$T(G-A,Opt) < T(G,Opt)$$

Proof: Follows directly from lemma 8.

In the following theorem we show that given any schedule S in which a processor is idle even though there is an available executable task, we can always find a strictly superior schedule S' in which a processor is idle only when there are no available executable tasks. We note that there does not exist an equally powerful theorem if task times are not random variables.

Theorem 3 A schedule on a graph in which task times are independent exponential random variables cannot be optimum if a processor is kept idle when there is an available executable task.

Proof: Let S be a schedule which initially assigns processors to tasks $\bar{a} = \{a_1,\ldots,a_k\}$ in G, and leaves some processor p idle even though there is an available executable task a_{k+1}. Assume that S follows the optimum policy after some task in $\bar{a}$ completes. Let S' be a schedule which makes the same assignment of processors to tasks as S, but in addition assigns processor p to task a_{k+1}.

From lemma 7,

$$T(G,Opt) > T(G-a_{k+1},Opt)$$

Hence for the given schedule S in G,

$$T(G,S) \geq T(G,Opt) > T(G-a_{k+1},Opt) \tag{61}$$

Let $\lambda = \sum_{i=1}^{k} \lambda_i$ and $\lambda' = \sum_{i=1}^{k+1} \lambda_i$

$$\text{Then } T(G,S) = \frac{1}{\lambda} + \sum_{i=1}^{k} \frac{\lambda_i}{\lambda} T(G-a_i,Opt) \tag{62}$$

$$\text{and } T(G,S') = \frac{1}{\lambda'} + \sum_{i=1}^{k+1} \frac{\lambda_i}{\lambda'} T(G-a_i,Opt) \tag{63}$$

From (61) and (62)

$$\frac{1}{\lambda} + \sum_{i=1}^{k} \frac{\lambda_i}{\lambda} T(G-a_i,Opt) > T(G-a_{k+1},Opt)$$

Multiplying both sides by $\lambda \cdot \lambda_{k+1}$, and then adding

$\lambda + \sum_{i=1}^{k} \lambda \lambda_i T(G-a_i,Opt)$ to both sides, we get

$$\lambda' + \sum_{i=1}^{k} \lambda'\lambda_i T(G-a_i,Opt) > \lambda + \sum_{i=1}^{k+1} \lambda \lambda_i T(G-a_i,Opt)$$

Dividing both sides by $\lambda\lambda'$ and using (62) and (63) gives

$$T(G,S) > T(G,S')$$

Summary of section 2 We have presented results which have no parallel in deterministic scheduling theory. We know that we should discard from consideration any schedule in which a processor is kept idle when there is an available task. We may (and shall) chose to discard from consideration schedules in which a processor is preempted at an instant when no other task is completed.

3. Properties of Exponential Forest Precedence Graphs

In this section we restrict attention to forest precedence graphs in which all task times are independent exponential random variables with unit means. Also we restrict attention to two processor schedules which may preempt only when a task completes.

We use the notation T(H,Z) to represent the expected time to complete all tasks in an EFPG, H, using schedule Z and assuming two identical processors. If H has only one executable task, then from (1) and (8),

$$T(H,Z) = \sum_i M_i = k_1 \tag{64}$$

Now consider the case where H has at least two available executable tasks. Since task times are independent exponential random variables with unit means it follows[5] that the expected time until the first of the two tasks processed by Z is completed is 1/2 and the probability that task z_i, i=1,2 finishes first is 1/2. Hence,

$$T(H,Z) = 1/2 + 1/2 \cdot T(H-z_1,Z) + 1/2 \cdot T(H-Z_2,Z) \tag{65}$$

Theorem 4

If G ~ H then T(G,X) = T(H,Y) where X and Y are arbitrary HLF schedules.

Proof: Let N and M be the number of tasks in G and H respectively. Since G ~ H it follows that N = M. The proof is an induction on N. For N = M = 0 we have T(G,X) = T(H,Y) = 0. Assume the theorem to be true for N = M = 0,1,..., P - 1 and we prove it true for N = M = P. Consider two cases: (1) G has only one executable task (2) G has two or more executable tasks.

Case 1. Here G has N tasks which must be executed in sequence. Since G ~ H, it follows that H also has N tasks which must be executed in sequence. Hence T(G,X) = T(H,Y) = N.

Case 2. Since G ~ H, it follows that H must also have at least two executable tasks, see (7) and (9). Hence from (65)

$$T(G,X) = \frac{1}{2} + \frac{1}{2} T(G - x_1,X) + \frac{1}{2} T(G - x_2,X) \tag{66}$$

$$T(H,Y) = \frac{1}{2} + \frac{1}{2} T(H - y_1,Y) + \frac{1}{2} T(H - y_2,Y) \tag{67}$$

From Lemma 3, $G - x_i \sim H - y_i$, for i = 1,2

Therefore, by the induction assumption,

$$T(G - x_i) = T(H - y_i),\ i = 1,2 \tag{68}$$

From (66) - (68), T(G,X) = T(H,Y)

If G and H are identical, then T(G,X) = T(G,Y). Hence all HLF schedules for the same EFPG have the same expected completion time. We shall use the notation T(G,HLF) to denote the expected completion time for G with 2 identical processors using <u>any</u> HLF schedule.

<u>Theorem 5</u>

Let G have at least two executable tasks. Then

$$j_1 < T(G,HLF) < N \tag{69}$$

Proof: By induction on N where N is the number of tasks in G. If G has at least two executable tasks and N = 2 then G must be a forest with two trees, each tree having one task; in other words, G has two independent tasks. In this case, i.e., N = 2, we have $j_1 = 1$ and T(G,HLF) = 1.5 and hence the theorem is satisfied. Assume the theorem to be true for N = 2,3,...,P-1 and we shall prove it true for N = P.

Consider two cases to prove the left part of inequality (69): (1) $N_i = 1$ for $i = 1,2,\ldots,j_1 - 1$ and $N_i = 2$ for $i = j_1$ (2) There either exists some $i < j_1$ such that $N_i > 1$, or $N_i = 1$ for $i = 1,2,\ldots,$ $j_1 - 1$ and $N_i > 2$ for $i = j_1$. In either case it is sufficient to show that for a given HLF schedule X,

$$j_1 < T(G,X) < N \tag{70}$$

Recall from (65) that

$$T(G,X) = \frac{1}{2} + \frac{1}{2} T(G - x_1,X) + \frac{1}{2} T(G-x_2,X) \tag{71}$$

Case 1. Here $N = j_1 + 1$, and $G - x_i$, $i = 1,2$ consists of $N - 1 = j_1$ tasks ordered in sequence.

Hence $T(G - x_i,X) = j_1$ for $i = 1,2$. Substituting in (71) yields

$$T(G,X) = j_1 + \frac{1}{2} > j_1$$

Case 2. Here $G - x_1$ must have at least two executable tasks, and furthermore $G - x_1$ must have at least one task in each level i, $i = 1,\ldots,j_1 - 1$. Hence

$$T(G - x_1,X) > j_1 - 1 \tag{72}$$

by the induction assumption.

$G - x_2$ must have at least one executable task at each level i, $i = 1,2,\ldots,j_1$. Obviously,

$$T(G - x_2,X) \geq j_1 \tag{73}$$

We now prove the right part of inequality (69). $G - x_i$ has $N - 1$ tasks, for $i = 1,2$. Hence

$$T(G - x_i,X) \leq N - 1 \text{ for } i = 1,2 \tag{74}$$

From (65) and (74)

$$T(G,X) \leq N - \frac{1}{2} < N$$

□

In theorem 6 we show that a flatter EFPG has a smaller expected time to complete all tasks. We note that a similar result does not exist in deterministic scheduling theory.

Theorem 6

If $G \prec H$ then $T(G,HLF) < T(H,HLF)$

Proof: By induction on M, the number of tasks in H. If $M = 1$ then $N = 0$ since $G \prec H$, and hence the theorem is trivially true. Asssume the theorem to be true for $M = 1,\ldots,P-1$ and we shall prove it true for $M = P$. It is

sufficient to show that for given HLF schedules X and Y, $T(G,X) < T(H,Y)$. Consider 4 cases: (1) Both G and H have at least two executable tasks. (2) Both G and H have less than two executable tasks (3) H has less than two executable tasks while G has at least two (4) G has less then two executable tasks while H has at least two.

Case 1. From Lemma 2 either $G - x_1 \prec H - y_1$ or $G - x_2 \prec H - y_2$. Therefore, by the induction assumption either:

$$T(G - x_1, X) < T(H - y_1, Y) \tag{75}$$

or $$T(G - x_2, X) < T(h - y_2, Y) \tag{76}$$

From Lemma 2 we have $G - x_i \prec H - y_i$, for both $i = 1$ and $i = 2$. Hence combining Theorem 4 and the induction assumption we have:

$$T(G - x_i, X) \leq T(H - y_i, Y) \quad i = 1,2 \tag{77}$$

From eqns (75) - (77) and (66), (67) we have

$$T(G,X) < T(H,Y)$$

Case 2. If G has zero executable tasks the theorem is trivially true. It is impossible for H to have zero executable tasks since $M = p-1 \geq 1$. Assume G and H have only one executable task each. Then G is a sequence of N completely ordered tasks and H is a sequence of M completely ordered tasks. Since $G \prec H$, it follows that $N < M$. Therefore

$$T(G,X) = N < M = T(H,Y)$$

Case 3. From Theorem 5

$$G(G,X) < N$$

Since $G \prec H$ it follows that $N \leq M$. Since H must have one executable task it follows that H is a sequence of M completely ordered tasks. Hence $T(H,Y) = M$. Therefore

$$T(G,X) < N \leq M = T(H,Y)$$

Case 4

By Theorem 5, $T(H,Y) > k_1$. If G has zero executable tasks the theorem is trivially true. Assume that G has one executable task. Then G is a completely ordered sequence of $j_1 = N$ tasks. Hence $T(G,X) = j_1$. Since $G \prec H$ it follows that $j_1 \leq k_1$. Therefore

$$T(G,X) = j_1 \leq k_1 < T(H,Y)$$

which completes the proof.

We summarize the results below:

From theorem 4, $G \sim H$ implies $T(G,HLF) = T(H,HLF)$ (78)

From theorem 6, $G \prec H$ implies $T(G,HLF) < T(H,HLF)$ (79)

From (78) and (79) $G \preceq H$ implies $T(G,HLF) \leq T(H,HLF)$ (80)

Theorem 7

A two-processor schedule on an EFPG is optimal if and only if it is HLF.

Proof: By induction on the number of tasks. If there is only one task, there is only one schedule, which is (trivially) an HLF schedule and the theorem follows.

If m = 2 then either G is a chain or G is a forest with two trees. If G is a chain then, again, there is only one schedule for G, and it is HLF. Likewise, if G is a forest there is only one schedule and it is HLF.

Assume the theorem to be true for all EFPGs with fewer than M tasks (M > 2). We show that the theorem holds for EFPGs with M tasks. Consider an arbitrary EFPG H with M tasks. If H is such that all schedules on H must be HLF schedules then the theorem is trivially true. Consider an EFPG H which allows two schedules Y and Z where Y is HLF and Z is <u>not</u> HLF.

$$T(H,Y) = \frac{1}{2} + \frac{1}{2} T(H - y_1,Y) + \frac{1}{2} T(H - y_2,Y)$$

$$T(H,Z) = \frac{1}{2} + \frac{1}{2} T(H - z_1,Z) + \frac{1}{2} T(H - z_2,Z)$$

Since Z is not HLF either (1) $\ell_1 < k_1$ or (2) $\ell_2 < k_2$ or Z is not HLF when applied to (3) $H - z_1$ or (4) $H - z_2$. The following results are required prior to proving each case. By Lemma 1, $H - y_i \precsim H - z_1$, $i = 1,2$. Therefore from (80),

$$T(H - y_i, \text{HLF}) \leq T(H - z_i, \text{HLF}), \quad i = 1,2 \tag{81}$$

By the induction assumption,

$$T(H - z_i, \text{HLF}) \leq T(H - z_i, Z), \quad i = 1,2$$

Hence

$$T(H - y_i, Y) \leq T(H - z_i, Z), \quad i = 1,2 \tag{82}$$

Case 1

If $\ell_1 < k_1$, then by Lemma 1, $H - y_1 \prec H - z_1$ and from eqn (79) and the induction assumption

$$T(H - y_1, Y) < T(H - z_1, Z)$$

Case 2

Similarly if $\ell_2 < k_2$ then

$$T(H - y_2, Y) < T(H - z_2, Z)$$

Cases 3 and 4

If Z is not HLF when applied to $H - z_i$, then by the induction assumption we have

$$T(H - z_i, \text{HLF}) < T(H - z_i, Z)$$

in which case by (81)

$$T(H - y_i, Y) < T(H - z_i, Z) \tag{83}$$

Hence in all 4 cases, (83) is true for either $i = 1$ or $i = 2$, and (82) is true for both $i = 1$ and $i = 2$. Hence $T(H,Y) < T(H,Z)$ which implies that $T(H,\text{HLF}) < T(H,Z)$. Therefore <u>all</u> HLF schedules have smaller expected completion times than any arbitrary non-HLF schedule. Hence non-HLF schedules cannot be optimal. Therefore <u>all</u> optimal schedules are HLF. Since <u>all</u> HLF schedules have the same completion time, <u>all</u> HLF schedules are optimal. This completes the proof.

This theorem is powerful because it states that any schedule which is not HLF cannot be optimal. Furthermore, any schedule which is HLF must be optimal. There is no parallel in deterministic theory in which the class of optimal schedules is completely characterized.

Up to this point we have considered two processor preemptive schedules. Now we demonstrate that preemption is not necessary by showing that there exists a non-preemptive HLF schedule for any arbitrary EFPG.

Lemma 9 Assume a graph G with a set $\bar{a}$ of k or more executable tasks. Then,

(1) If there are k or more executable tasks in G - a then there exists a set $\bar{A}$ of the k highest level tasks which includes $\bar{a} - a_j$

(2) If there are only k-1 executable tasks in G - a_j then this set $\bar{A}$ of executable tasks is $\bar{a} - a_j$.

Proof: (1) Assume the contrary, i.e. that there exist at least k executable tasks in G - a_j and there does not exist any set $\bar{A}$ of the k highest level tasks in G - a_j which includes $\bar{a} - a_j$. If there is only 1 executable task b, not in $\bar{a} - a_j$, with a level greater than some task in $\bar{a} - a_j$, then set $\bar{A} = b \cup \bar{a} - a_j$ which leads to a contradiction!

Therefore, assume there are $p \geq 2$ tasks in G - a_j with a level greater than the level of some task a_i in $\bar{a} - a_j$. If b_r, $1 \leq r \leq p$ was an executable task in G then since the level of b_r exceeds that of a_i, it follows that $\bar{a}$ is not a set of the k highest level tasks in G. Contradiction! Hence b_r, $1 \leq r \leq p$ cannot be executable in G. Since we have assumed $\forall_r$ that b_r is executable in G - a_j, it follows that $\forall_r$ b_r must have a_j for a predecessor in G, which implies that a_j has $p \geq 2$ successors and hence G is not a forest. Contradiction!

(2) By assumption there are only k-1 executable tasks in G - a_j and $\bar{a} - a_j$ is a set of k - 1 executable tasks in G - a_j and hence every executable task is in $\bar{a} - a_j$.

Theorem 8 There exists a non-preemptive HLF schedule for any arbitrary EFPG G.

Let there be k processors. Consider 2 cases. (1) There are k or fewer executable tasks (2) There are more than k executable tasks.

Case 1: Since G is a forest, there will be k or fewer executable tasks in every subgraph of G. Hence any schedule in which every executable task is assigned a free processor is HLF and non-preemptive.

Case 2: Assign processors to a set of k highest-level tasks $\bar{a} = \{a,\ldots,a_k\}$. Let a_j be the first task to complete. Choose a new set a' of k highest-level tasks where a' includes $\bar{a} - a_j$; note that from the previous lemma such a set always exists. Suppose $a' = (\bar{a} - a_j) \cup b$. Do not change the assignment of processors to $\bar{a} - a_j$ but merely assign the processor which was assigned to a_j to b. We now have achieved a transition from an HLF assignment of processors to tasks in G to an HLF assignment of processors to tasks in $G - a_j$ without preempting any processor. We now apply the same method to assign processors for all subgraphs that result in processing $G - a_j$.

Corollary 1: Even if zero-delay preemption were permissible, there exists an optimum non-preemptive (HLF) schedule for 2 processor schedules on EFPGs; in other words preemption provides no advantage! Once again, it is instructive to contrast this result with deterministic theory where preemption may provide an advantage [8,9]

4. Bounds for Scheduling when Task Times Are Unknown.

In section 5 we present a number of configurations of processors and probabilistic task graphs for which HLF schedules are not necessarily optimal. At this time there are no known optimal scheduling algorithms for the general cases of the examples we give. However, we have found that it is possible to place a bound on the ratio between an optimal scheduling algorithm, whatever it may be, and an arbitrary scheduling algorithm, for certain configurations of processors and task graph types.

Consider the scheduling algorithms for a given deterministic task graph and m identical processors where the algorithms are non-preemptive and never leave a processor idle if a task is available. Among these algorithms is one which is (not necessarily exclusively) optimal for the given task graph. Call this schedule "0". Graham [6] has shown that the relation between the completion time, ω_0, using 0, and the completion time, ω_a, using an arbitrary schedule "a", is

$$\omega_a \leq (2 - 1/m)\, \omega_0 \tag{84}$$

Now we consider the case where task times are selected from a probability distribution. For a given instantiation of a probabilistic task graph we consider the theoretical optimal schedule "0" given the task execution times in advance. Of course since it is not possible to know task execution times in advance we can never determine 0 in advance in practice. However, we shall use 0 in some "thought" experiments. Also we consider an arbitrary schedule "a" which may be derived independent of the task execution times. For any particular instantiation of a task graph, the derived schedule "0", and any schedule "a", Graham's bound still applies.

Over the space of all possible instantiations of a probabilistic

task graph, ω_0 and ω_a are random variables. While the distributions for ω_0 and ω_a can be obtained from the distributions of the instantiations of a given probabilistic task graph, we restrict our attention to the mean values for ω_0 and ω_a.

Because ω_a and ω_0 are random variables we may substitute expected values for them into the inequality (84):

$$E[\omega_a] \leq (2 - 1/m) \cdot E[\omega_0] \tag{85}$$

The interpretation of this inequality is that the expected value of the completion time for an arbitrary schedule is not greater than (2 - 1/m) times the expected value of completion times for optimal schedules given task times in advance.

We now consider another schedule "b". Assuming ω_b is the completion time for "b" for a given instantiation of a probabilistic task graph, then for any given instantiation

$$\omega_b \geq \omega_0$$

Hence

$$E[\omega_b] \geq E[\omega_0] \tag{86}$$

From (85) and (86), given arbitrary schedules "a" and "b" and any precedence graph with probabilistic task times,

$$E[\omega_a] \leq (2-1/m) \quad E[\omega_b]$$

The significance of (87) is twofold. First,it places bounds on the relationship between two arbitrary schedules. Essentially, no pair of schedules can produce finishing times that, on the average, differ by more than a factor of (2 - 1/m) when task times are probabilistic.

Second, and perhaps more important, (86) implies that we might expect the factor (2 - 1/m) to diminish in (87). For example, despite the fact HLF is optimal for EFPGs on the average, it is easy to construct forest

precedence graphs with fixed task times which clearly make HLF a suboptimal algorithm. Thus, due to randomness, we can expect even an optimal algorithm such as HLF and an arbitrary algorithm to be more closely related than (2 - 1/m). This observation has been borne out in empirical studies reported in [1].

5. Extensions

In this section we investigate cases which represent extensions to the restrictions assumed in section 3. For example, what happens if we allow three processors with EFPGs, or two processors but allow any type of precedence graph? Section 5.1 presents two cases in which the results of section 3 are extendable. Remaining sections demonstrate that "simple" modifications to EFPGs or to the two processors assumption can nullify the optimality of the HLF algorithm.

For the latter cases we have not found any optimal polynomial algorithms nor have we established NP-completeness [10].

5.1 EFPGs with two non-identical processors

5.1.1 The preemptive case

The results of section 3 can be extended to the case where the two processors have different rates. The proofs for this case are an extension of the proofs in section 3 and are not presented here. Instead we present an outline of the differences created by assuming non-identical processors.

If the mean execution time for tasks assigned to the fast processor is unity then we can represent the time for tasks on the slow processor as $1/a$, where $a \leq 1$. If, for schedule X, the fast processor is assigned task x_1 in EEPG G and the slow processor is assigned x_2, then (65) is modified to

$$T(G,x) = \frac{1}{1+a} + \frac{1}{1+a} T(G-x_1,X) + \frac{a}{1+a} T(G-x_2, X)$$

A schedule is defined to be HLF if and only if, at all times the fast processor is assigned an executable task at a higher level than all other executable tasks and the slow processor is then assigned an executable task at the highest level among the remaining executable tasks. Thus, a task assigned to the slow processor may be preempted and assigned to the fast processor, resulting in a maximum of N - 1 task preemptions.

5.1.2 The non-preemptive case

It has been shown for non-preemptive scheduling of some deterministic models that keeping a processor idle even when there is a ready task can produce smaller finishing times [6]. The same result applies to EFPGs when there are two or more non-identical processors. A two processor example is given in Figure 5 where, initially, the slow processor is assigned to task 1 and the fast processor is assigned to task 2 (Fig. 5a). If the fast processor completes its task first then it is assigned next to task 3 (Fig. 5b). Assuming that the slow processor completes task 1 next, it now becomes possible, given widely differing processor speeds, that it would be advantageous to not assign the slow processor to task 4 but rather to wait for the fast processor to complete task 3 so that it could then be assigned to task 4, and subsequently the remainder of the chain (Fig. 5c).

5.2 Three processor EFPG schedules

The optimal three processor schedule for the EFPG shown in figure 4 is not an HLF schedule; tasks 1, 2 and 4 are processed initially in the optimal schedule whereas tasks 1,2 and 3 are processed initially in the HLF schedule. This fact (i.e. that 1,2 and 4 are the optimum selection of tasks initially) is not obvious; it can only be shown by laboriously computing mean execution times for the different schedules, and we have written a program to do so.

The advantages of flatness as established in section 3 do not hold in this case. If H has 3 executable tasks and $G \prec H$, and G and H have the same number of tasks; it is possible that G has only two executable tasks (Figs. 1,3). Hence, given three processors H offers more opportunity for exploiting parallelism and as a result has a smaller expected completion time.

5.3 Precedence Graphs which are not forests

The optimal two-processor schedule for the graph shown in Fig. 6 in which task times are iid exponential is not HLF; the optimal solution is to process tasks 1 and 3 initially whereas the HLF schedule processes tasks 1 and 2 initially. The reason that the HLF algorithm is not optimal in this case is that task 3 has many more successors than task 2; hence completing task 3 opens up more opportunity for parallelism.

5.4 Forests with unequal exponentially distributed task times

The optimal two processor schedule for the forest precedence graph shown in Fig. 7 is not an HLF schedule. An optimal solution in this case is to first schedule tasks 1 and 2 and then to schedule the remaining task with task 3. An HLF schedule would have selected task 3 and either task 1 or task 2 initially.

6.0 Conclusion

Until now the polynomial time heuristics for optimally scheduling partially ordered tasks have required that task times be known or known to be equal. The successful relaxing of this constraint to iid exponential distributions is a first indication that it may be possible to derive other results for partially ordered tasks with probabilistic task times. For example, there is evidence that HLF schedules may be optimal for forest precedence graphs where task times are iid distributions in which the expected time to completion for a task is a monotonically non-increasing function of the amount of processing received so far. Proofs for these conditions have not been completed by the authors.

Examples given in section 5 (namely, 5,2,5,3,5,4) demonstrate that results in deterministic scheduling, such as those found in [8,9] do not necessarily generalize to the probabilistic task time case. We expect in these cases that optimal algorithms will be found to be NP-complete.

The power of theorems obtained in the stochastic case relative to those in the deterministic case are worth noting . It is indeed fascinating to note that any optimal algorithm with 2 processors must be HLF, while HLF algorithms may be suboptimal with 3 processors. There appears to be no characterization of optimal schedules that applies to 2 and 3 processor cases.

In terms of practical applications the extension of Graham's bound has interesting implications. As observed in the proof of its applicability to probabilistic task times we can expect, due to randomness, that the average completion times for two schedules will differ by a factor less than 2 - 1/number of processors. Also, we would expect an arbitrary schedule for a two processor system to be less than 50% worse than an

optimal schedule. Results in [1] indicate this percentage is more likely to be less than 15% for an arbitrary algorithm and less than 2% for HLF.

Thus, even though HLF is demonstrably inferior in some cases, its average behavior is very good.

References

1. Adam, T. L., Chandy, K. M. and Dickson, J. R., A Comparison of List Schedules for Parallel Processing Systems. Comm. ACM, 17, 12, (1974), 685-691.

2. Coffman, E. G. and Denning, P. J., Operating Systems Theory, Prentice-Hall, Englewood Cliffs, N. J. (1974).

3. Coffman, E. G. and Graham, R. L. Optimal Scheduling for Two-processor Systems. Acta Informatica, 1.3 (1972), 200-213.

4. Drake, A. W., Fundamentals of Applied Probability Theory, McGraw-Hill Book Company, New York, 1967.

5. Feller, W., An Introduction to Probability Theory and Its Applications, vol. 1 2d ed., John Wiley and Sons, Inc., New York, 1957.

6. Graham, R. L., "Bounds on Multiprocessing Timing Anomalies", SIAM J. Appl. Math, 17, 2 (1969), 416-440.

7. Hu, T. C. Parallel Sequencing and Assembly Line Problems, Operations Research, 9, (Nov-Dec 1961), 841-848.

8. Muntz, R. R. and Coffman, E. G. Optimal Preemptive Scheduling on Two-processor Systems, IEEE Trans. C-18, 11, (Nov 1969), 1014-1020.

9. Muntz, R. R. and Coffman, E. G. Preemptive Scheduling of Real Time Tasks on Multiprocessor Systems. Journal ACM, 17, 2, (1970), 324-338.

10. Ullman, J. D., "Polynomial Complete Scheduling Problems", Proc. 4th Symposium on Operating Systems Principles (1973), 96-101.

i	N_i	S(G,i)
7	1	1
6	1	2
5	1	3
4	2	5
3	2	7
2	2	9
1	1	10

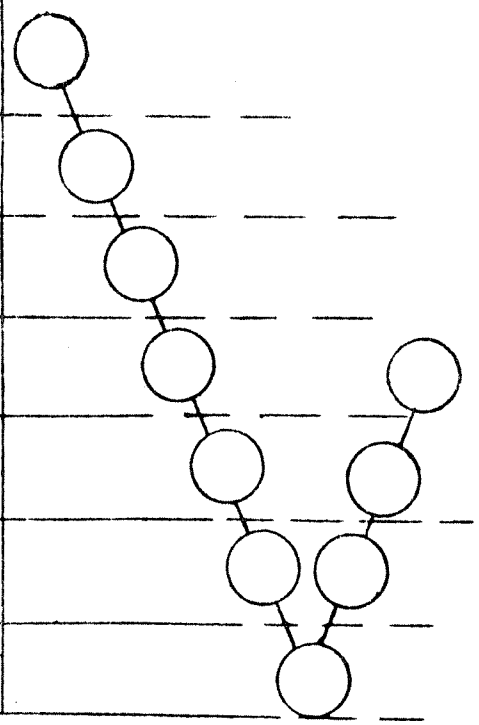

Figure 1.

i	N_i	S(G,i)
7	1	1
6	1	2
5	2	4
4	1	5
3	2	7
2	2	9
1	1	10

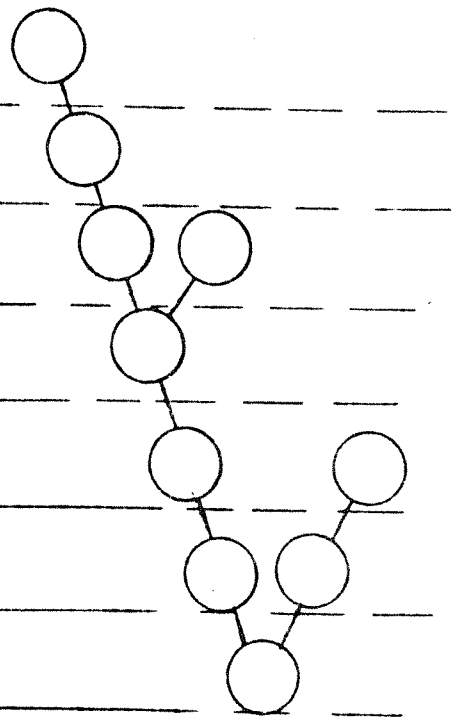

Figure 2.

i	N_i	S(G,i)
7	1	1
6	1	2
5	1	3
4	2	5
3	2	7
2	2	9
1	1	10

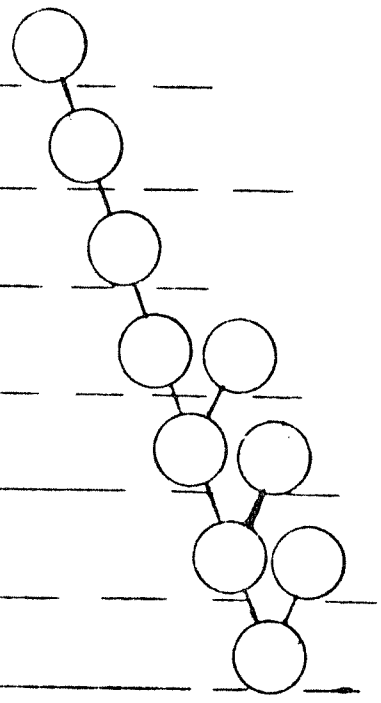

Figure 3.

i	N_i	S(G,i)
7	2	2
6	1	3
5	2	5
4	2	7
3	2	9
2	2	11
1	1	12

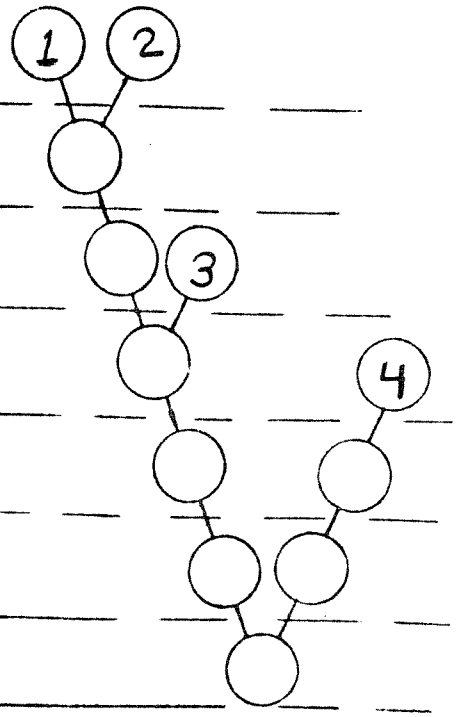

Figure 4.

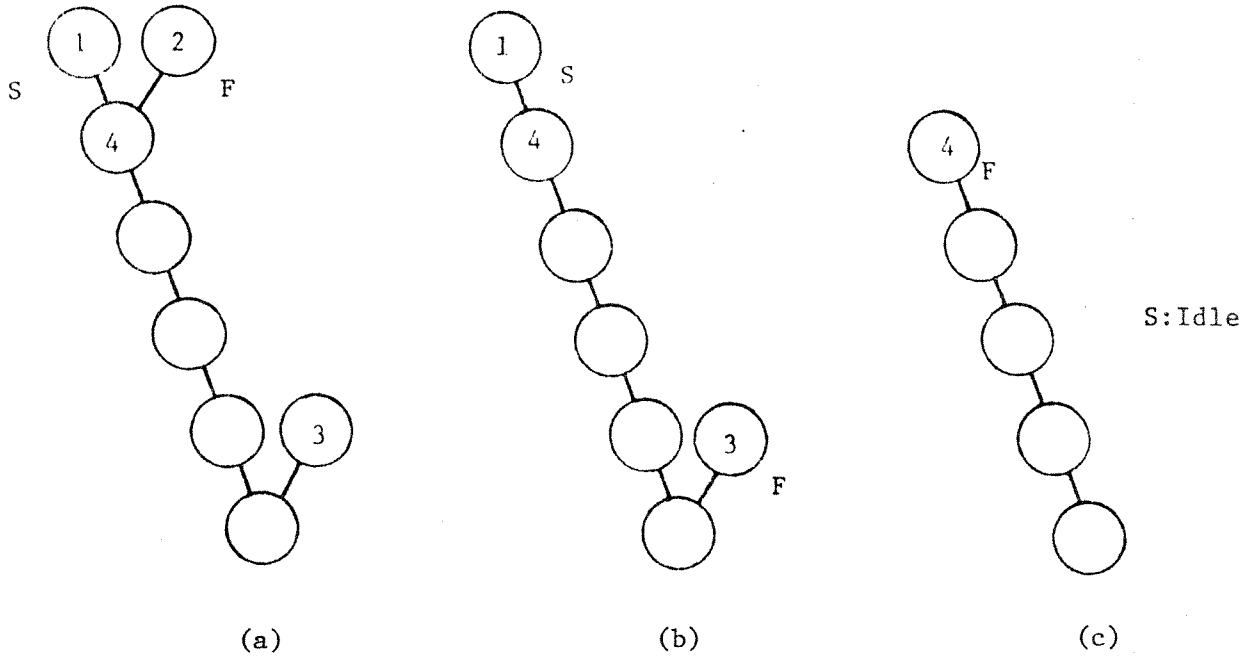

Figure 5. Keeping a Slow Processor Idle.

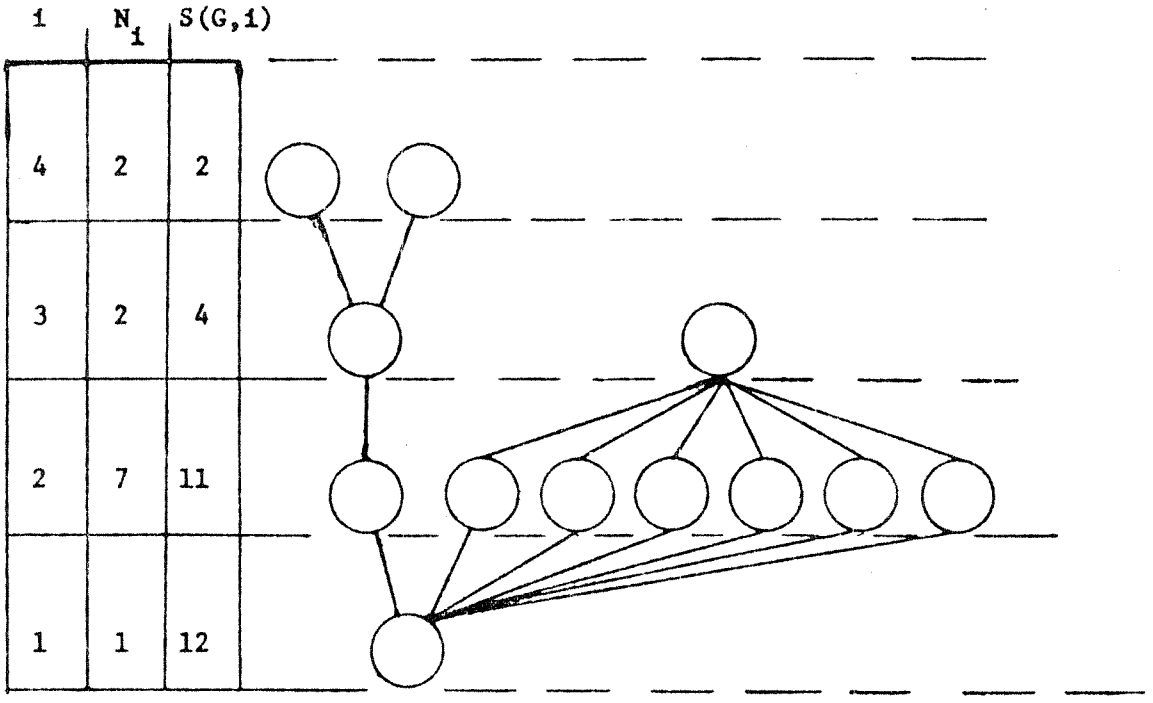

Figure 6.

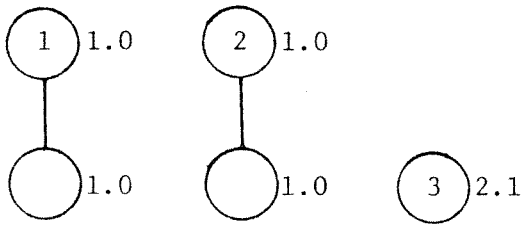

Figure 7. Forest with Unequal Exponentially Distributed Task Times.