Evaluating the Mean Completion Time of a Fork-Join Barrier Synchronization

John Strikwerda
Rajesh K. Mansharamani

Technical Report #1153

May 1993

Evaluating the Mean Completion Time of a Fork-Join Barrier Synchronization *

John Strikwerda and Rajesh K. Mansharamani
strik@cs.wisc.edu *mansha@cs.wisc.edu*
Computer Sciences Department
University of Wisconsin-Madison
1210 W. Dayton St.
Madison, WI 53706.

May 20, 1993

Abstract

In this paper we analyze the mean completion time of a fork-join barrier synchronization for a parallel program model proposed in previous literature. The task service times in the model are derived from the total job demand by ratios of uniform random variables. We first compute an analytic expression for the mean time to barrier completion, $S(n)$, for jobs that have n tasks. This expression is an alternating series of combinatorial terms that is numerically unstable for large values of n if summed in straightforward manner. We next derive an alternate method of computing $S(n)$ that consists of an infinite series of positive terms that sums to less than 1. We obtain error estimates for truncated sums of this series and using the error estimates compute $S(n)$ to a desired level of accuracy. Finally, we compare estimates of the mean time to barrier completion with other common models of parallel program task execution times. The main emphasis of this paper is on the techniques used to derive and compute $S(n)$.

*This research was partially supported by the National Science Foundation under grants DMS-9208049 and CCR-9024144.

1 Introduction

Analytic models for parallel programs often assume programs to be of the fork-join type where a program forks into n tasks that synchronize (join) at a barrier. The time to completion of a single barrier is the time for the last task to reach the barrier. To compute the mean time for barrier synchronization we need to know how total job demand[1] is divided among the tasks of the fork-join program. One model of division of total demand amongst tasks is as follows. Consider a fork-join program with n tasks and total job demand D. The service requirement of the jth task, $j = 1, 2, \ldots, n$ is given by the random variable

$$T_j = \frac{U_j}{\sum_{i=1}^{n} U_i} D, \tag{1}$$

where $U_1, \ldots, U_n$ are independent and identically distributed random numbers uniformly distributed between 0 and 1. This model has been used in two previous simulation studies of the performance of multiprogrammed parallel processor allocation policies [5, 3]. Note that the randomness in the above division of demand reflects the fact that different programs with n tasks will in general have different divisions of demand amongst their tasks, rather than randomness in task service times. In fact, a recent paper by Adve and Vernon [1] makes a strong case that for a given parallel program one can essentially treat task service times as deterministic and include just the mean communication overhead into the task service time. In the above model this is represented by fixing the values of U_j, $j = 1, 2, \ldots, n$ for a given program.

The mean service time (barrier completion time) of a fork-join program with n tasks is given by

$$S(n) = E[\max\{T_1, \ldots, T_n\}], \tag{2}$$

where we assume that all tasks execute in parallel, each on its own processor. The simulation studies mentioned above did not analytically compute the mean service time for the model given by (1) because they were interested in computing the mean response time of a multiprogrammed system under certain workload conditions. In this paper we are concerned with the problem of how to analytically compute the mean service time of the program model given by (1). This will yield insight into how easy is it to analyze the model for performance evaluation of a multiprogrammed system. Moreover, estimates of $S(n)$ will also determine how well this model compares to the best case of linear speedups under which $S(n) = D/n$, and

[1] We refer to job demand as the total execution time on a single processor.

how well it compares to other models for a mix of parallel programs.

We first compute an analytic expression for $S(n)$ using the model given by (1). This expression is an alternating series of combinatorial terms and is numerically unstable for large values of n if summed in straightforward manner. We next derive an alternate method of computing the same expression for $S(n)$ that is numerically stable for large n since it consists only of positive terms rather than alternating terms. This derivation involves techniques that should be of interest to numerical analysts and mathematicians. Using this alternate derivation we compute the mean barrier completion times as a function of the number of tasks in the program (assuming that each task runs on its own processor). We also discuss how the estimates of mean barrier completion times compare with other models of program behavior.

The rest of this paper is organized as follows. Section 2 derives an analytic expression for $S(n)$. Section 3 demonstrates the limitations of a straightforward computation of $S(n)$. In Section 4 we derive an alternate way to compute the same expression for $S(n)$. Section 5 studies the numerical stability of the alternate method, and finally Section 6 plots $S(n)$ versus n using the derivation from Section 4 and then compares the estimates of $S(n)$ with two other models of task service times..

2 Analytic Expression for $S(n)$

In this section we compute $S(n)$ given by equation (2) where the task service times T_j, $j = 1, 2, \ldots, n$ are given by equation (1). For the sake of convenience we set total job demand $D = 1$, which means that $0 \leq S(n) \leq 1$. By definition we have

$$\begin{aligned} S(n) &= \int_0^1 \int_0^1 \cdots \int_0^1 \max\left(\frac{u_1}{\sum_{i=1}^n u_i}, \cdots, \frac{u_n}{\sum_{i=1}^n u_i}\right) \; du_1 \ldots du_n \\ &= \int_0^1 \int_0^1 \cdots \int_0^1 \frac{\max(u_1, \ldots, u_n)}{\sum_{i=1}^n u_i} \; du_1 \ldots du_n \\ &= n \int_0^1 \int_0^{u_n} \cdots \int_0^{u_n} \frac{u_n}{\sum_{i=1}^n u_i} \; du_1 \ldots du_{n-1} du_n. \end{aligned} \tag{3}$$

The last equation follows by symmetry of the integrand in the n dimensional hypercube $[0,1]^n$ (each of $u_1, \ldots, u_n$ is maximum in a particular region of $[0,1]^n$ and the value of the integral is the same in each such

region.) For example, when $n = 2$ we have

$$S(2) = \int_0^1 \int_0^1 \frac{\max(u_1, u_2)}{u_1 + u_2}\, du_1 du_2 \quad = 2\int_0^1 \int_0^{u_2} \frac{u_2}{u_1 + u_2}\, du_1 du_2 \quad = 2\int_0^1 \int_{u_2}^1 \frac{u_1}{u_1 + u_2}\, du_1 du_2.$$

We will use equation (3) to obtain the following expression for $S(n)$.

Theorem 2.1

$$S(n) = \frac{1}{(n-2)!} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^i (n-i)^{n-2} \ln(n-i) \quad n = 2, 3, \ldots \tag{4}$$

Note that $S(1) = 1$ by definition (since $D = 1$).

In order to prove Theorem 2.1 and also for the purposes of Section 4 we will make extensive use of the following proposition.

Proposition 2.1 *For any real numbers a, b, and c*

$$\sum_{i=0}^{m} \binom{m}{i} (-1)^i (a + b(c-i))^\ell = 0, \quad \ell = 0, 1, \ldots, m-1. \tag{5}$$

Proof.

$$\begin{aligned}
\sum_{i=0}^{m} \binom{m}{i} (-1)^i (a + b(c-i))^\ell &= \sum_{i=0}^{m} \binom{m}{i} (-1)^i \left[\frac{d^\ell}{dx^\ell} \left\{ e^{(a+b(c-i))x} \right\} \Big|_{x=0} \right] \\
&= \frac{d^\ell}{dx^\ell} \left[e^{(a+bc)x} \sum_{i=0}^{m} \binom{m}{i} (-1)^i e^{-bx\, i} \right]_{x=0} \\
&= \frac{d^\ell}{dx^\ell} \left[e^{(a+bc)x} \left(1 - e^{-bx}\right)^m \right]_{x=0} \\
&= 0, \quad \ell = 0, 1, \ldots, m-1.
\end{aligned}$$

The last equality is true because the ℓth derivative of the expression ($\ell < m$) has $(1 - e^{-bx})^{m-\ell}$ as a factor, which evaluates to zero. ∎

We will also make use of the following equation (which is obtained by integration by parts).

$$\int (a + bx)^j \ln(a + bx)\, dx = \frac{(a + bx)^{j+1}}{b(j+1)} \left(\ln(a + bx) - \frac{1}{j+1} \right). \tag{6}$$

We are now in a position to prove Theorem 2.1 which follows almost immediately from the following Lemma. We will use the notation $\sigma_{k,n}$ to denote $u_k + u_{k+1} + \ldots + u_n$, $1 \leq k \leq n$ (where $\sigma_{n,n} = u_n$).

Lemma 2.1 *Let*

$$X_j(u_n) = \int_0^{u_n} \cdots \int_0^{u_n} \frac{u_n}{\sigma_{1,n}} \, du_1 du_2 \ldots du_j, \quad j = 1, 2, \ldots, n-1.$$

Then

$$X_j(u_n) = \frac{u_n}{(j-1)!} \sum_{i=0}^{j} \binom{j}{i} (-1)^i (\sigma_{j+1,n} + (j-i)u_n)^{j-1} \ln(\sigma_{j+1,n} + (j-i)u_n), \quad j = 1, 2, \ldots, n-1. \quad (7)$$

Proof. We prove this Lemma by induction on j.

1. *Basis step:* Consider $j = 1$. By definition

$$X_1(u_n) = \int_0^{u_n} \frac{u_n}{\sigma_{1,n}} du_1 = [u_n \ln \sigma_{1,n}]_{u_1=0}^{u_1=u_n} = u_n(\ln(\sigma_{2,n} + u_n) - \ln \sigma_{2,n}),$$

   which satisfies (7).

2. *Induction Hypothesis:* Assume (7) to hold for $j = 1, 2, \ldots, k$, $k < n-1$.

3. *Show that (7) holds for $j = k+1$:* We have

$$\begin{aligned} X_{k+1}(u_n) &= \int_0^{u_n} X_j(u_n) du_{k+1} \\ &= \int_0^{u_n} \frac{u_n}{(k-1)!} \sum_{i=0}^{k} \binom{k}{i} (-1)^i (\sigma_{k+1,n} + (k-i)u_n)^{k-1} \ln(\sigma_{k+1,n} + (k-i)u_n) \; du_{k+1} \\ &= \frac{u_n}{(k-1)!} \sum_{i=0}^{k} \binom{k}{i} (-1)^i \int_0^{u_n} (\sigma_{k+1,n} + (k-i)u_n)^{k-1} \ln(\sigma_{k+1,n} + (k-i)u_n) \; du_{k+1} \end{aligned}$$

   We evaluate this integral using equation (6) (set $a = \sigma_{k+1,n}$, $b = (k-i)$, $x = u_n$, and $j = k-1$) to get

$$\begin{aligned} X_{k+1}(u_n) &= \frac{u_n}{(k-1)!} \sum_{i=0}^{k} \binom{k}{i} (-1)^i \left[\frac{(\sigma_{k+1,n} + (k-i)u_n)^k}{k} \left(\ln(\sigma_{k+1,n} + (k-i)u_n) - \frac{1}{k} \right) \right]_{u_{k+1}=0}^{u_{k+1}=u_n} \\ &= \frac{u_n}{k!} \sum_{i=0}^{k} \binom{k}{i} (-1)^i \left[(\sigma_{k+2,n} + (k+1-i)u_n)^k \left(\ln(\sigma_{k+2,n} + (k+1-i)u_n) - \frac{1}{k} \right) \right. \; - \end{aligned}$$

$$(\sigma_{k+2,n} + (k-i)u_n)^k \left(\ln(\sigma_{k+2,n} + (k-i)u_n) - \frac{1}{k}\right)\Bigg]$$

$$= \frac{u_n}{k!}\sum_{i=0}^{k}\binom{k}{i}(-1)^i\left[f_{i-1} - f_i\right],$$

where

$$f_i = (\sigma_{k+2,n} + (k-i)u_n)^k \left(\ln(\sigma_{k+2,n} + (k-i)u_n) - \frac{1}{k}\right).$$

The above readily simplies to

$$\begin{aligned}
X_{k+1}(u_n) &= \frac{u_n}{k!}\left[f_{-1} + \sum_{i=0}^{k-1}(-1)^{i+1}\left\{\binom{k}{i+1} + \binom{k}{i}\right\}f_i - (-1)^k f_k\right] \\
&= \frac{u_n}{k!}\left[f_{-1} + \sum_{i=0}^{k-1}(-1)^{i+1}\binom{k+1}{i+1}f_i + (-1)^{k+1}f_k\right] \\
&= \frac{u_n}{k!}\left[f_{-1} + \sum_{i=1}^{k}(-1)^{i}\binom{k+1}{i}f_{i-1} + (-1)^{k+1}f_k\right] \\
&= \frac{u_n}{k!}\sum_{i=0}^{k+1}(-1)^i\binom{k+1}{i}f_{i-1} \\
&= \frac{u_n}{k!}\sum_{i=0}^{k+1}(-1)^i\binom{k+1}{i}(\sigma_{k+2,n} + (k+1-i)u_n)^k\left(\ln(\sigma_{k+2,n} + (k+1-i)u_n) - \frac{1}{k}\right)
\end{aligned}$$

Comparing this equation with (7) for $j = k+1$, we note that the only difference is that the $-1/k$ term is missing from (7). Hence it remains to show that

$$-\frac{1}{k}\sum_{i=0}^{k+1}(-1)^i\binom{k+1}{i}(\sigma_{k+2,n} + (k+1-i)u_n)^k = 0$$

This readily follows from Proposition (2.1) by setting $m = k+1$, $a = \sigma_{k+2,n}$, $b = u_n$, $c = k+1$, and $\ell = k < k+1$ in the Proposition.

∎

The proof of Theorem 2.1 easily follows from Lemma 2.1.

Proof of Theorem 2.1:

From equation (3) and Lemma 2.1 we have

$$
\begin{aligned}
S(n) &= n\int_0^1 X_{n-1}(u_n)\,du_n \\
&= n\int_0^1 \frac{u_n}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(\sigma_{n,n}+(n-1-i)u_n)^{n-2}\ln(\sigma_{n,n}+(n-1-i)u_n)\;\;du_n \\
&= \frac{n}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i\int_0^1 u_n((n-i)u_n)^{n-2}\ln((n-i)u_n)\;\;du_n \\
&= \frac{n}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(n-i)^{n-2}\int_0^1 u_n^{n-1}(\ln u_n+\ln(n-i))\;\;du_n \\
&= \frac{n}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(n-i)^{n-2}\left[\frac{u_n^n}{n}\left(\ln u_n-\frac{1}{n}\right)+\frac{u_n^n}{n}\ln(n-i)\right]_{u_n=0}^{u_n=1} \\
&= \frac{n}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(n-i)^{n-2}\left[-\frac{1}{n^2}+\frac{1}{n}\ln(n-i)\right] \\
&= \frac{1}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(n-i)^{n-2}\ln(n-i)-\frac{1}{n\,(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(n-i)^{n-2} \\
&= \frac{1}{(n-2)!}\sum_{i=0}^{n-1}\binom{n-1}{i}(-1)^i(n-i)^{n-2}\ln(n-i)-0
\end{aligned}
$$

The last summation evaluates to zero by virtue of Proposition 2.1 (set $m=n-1$, $a=0$, $b=1$, $c=n$, and $\ell=n-2<n-1$). ∎

3 Numerical Instability of Straightforward Summations

This section explores two straightforward ways of summing the right hand side of equation (4). We show that both these methods are numerically unstable for $n > 25$. In both cases the instability results from subtracting two large numbers that are almost equal to each other resulting in significant loss of precision. In fact, both methods produce negative values for $S(30)$ which is clearly impossible since $S(n) > 0, \quad \forall n$.

To begin with we note from (4) that $S(n)$ simplifies to

$$\begin{aligned} S(n) &= (n-1)\sum_{i=0}^{n-1} \frac{1}{i!\,(n-1-i)!}(-1)^i (n-i)^{n-2} \ln(n-i), \qquad n > 1 \\ &= (n-1)\sum_{i=0}^{n-1} (-1)^i \ln(n-i) \prod_{j=1}^{i} \left(\frac{n-i}{j}\right) \prod_{j=2}^{n-1-i} \left(\frac{n-i}{j}\right) \end{aligned}$$

The first method of summation (called A1) that we examine does precisely what the above expression states. That is,

$$\textbf{A1:} \qquad S(n) = (n-1)\sum_{i=0}^{n-1} (-1)^i f(i),$$

where

$$f(i) = \ln(n-i) \prod_{j=1}^{i} \left(\frac{n-i}{j}\right) \prod_{j=2}^{n-1-i} \left(\frac{n-i}{j}\right).$$

We computed $S(n)$ using method A1 on a DECstation 3100. The method was coded in C using double precision arithmetic. Figure 1 depicts the values of $S(n)$ that were obtained using this method. Note the anomalous behavior of $S(n)$ after $n = 25$. By definition $S(n)$ is nonincreasing and positive (see equations (1) and (2)). However, Figure 1 shows that $S(29) > S(28) > S(26) > S(25)$, and furthermore $S(30) < 0$. This clearly reveals that method A1 is numerically unstable after $n = 25$.

To obtain insight into the numerical instability of this method we note that the computation proceeds with $(-1)^i f(i)$ being added to the partial sums at every step i. If the partial sum is close to $f(i)$ in magnitude then this will result in loss of significance if the partial sum up to step i has an opposite sign to that of $(-1)^i f(i)$. A closer examination of the partial sums obtained using method A1 confirms our conjecture. Table 1 tabulates some of the partial sums $s(k-1)$ versus $(-1)^k f(k)$ for $n = 30$ where $s(k-1) = \sum_{i=0}^{k-1} (-1)^i f(i)$. This partial sum is added to $(-1)^k f(k)$ at the k^{th} step. From the table we see that the partial sums $s(k-1)$ are quite close to $(-1)^k f(k)$ in magnitude but opposite in sign for $k < 25$ and the cumulative effect of the loss of significance up to $k = 25$ shows its impact after $k = 25$ when the values of $f(k)$ are almost zero. The loss of significance due to the previous additions has caused the partial sum at $k = 25$ to take on a negative value of approximately -0.05 which causes the resultant value of $S(30)$ to be negative. We therefore need a better way of summing (4).

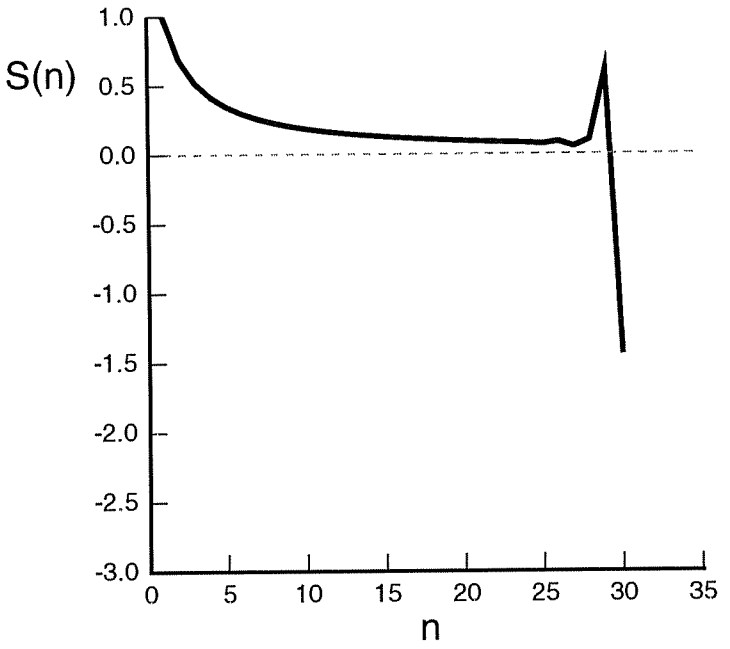

Figure 1: $S(n)$ versus n: Method A1

k	$s(k-1)$	$(-1)^k f(k)$
1	8.800111e+10	-9.778774e+11
5	2.429620e+13	-5.999811e+13
10	-1.273704e+13	1.821735e+13
15	1.796136e+10	-2.024436e+10
20	-2.553012e+04	2.608123e+04
25	-4.971860e-02	-1.610562e-07
26	-4.971876e-02	4.128240e-11
27	-4.971876e-02	-1.154056e-15
28	-4.971876e-02	6.102735e-22
29	-4.971876e-02	-0.000000e+00

Table 1: Comparison of partial sums and sequence terms for $n = 30$: Method A1

Since the previous method of summation caused errors due to the addition of numbers that were close to each other in magnitude but opposite in sign, the second method that we examine (method A2) simply adds the positive terms of the series first, then the negative terms and finally adds these two summands. More precisely, we have

$$\textbf{A2:} \qquad S(n) = (n-1)\left\{ \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} f(2i) - \sum_{i=0}^{\lfloor (n-1)/2 \rfloor - 1} f(2i+1) \right\},$$

where $f(i)$ is as defined before. Figure 2 plots $S(n)$ versus n as computed by method A2 on a DECstation 3100 using double precision arithmetic (the method was coded in C). We see that method A2 fares no better than method A1. The reason for this is as follows. By definition $0 < S(n) < 1$, $n > 1$ (since $D = 1$ in equation (1)). Consider the values of the summands in method A2, that is, $s_e(n) = \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} f(2i)$ and $s_o(n) = \sum_{i=0}^{\lfloor (n-1)/2 \rfloor - 1} f(2i+1)$, corresponding to sum of even terms and sum of odd terms respectively. Table 2 displays $s_e(n)$ and $s_o(n)$ for various values of n, and we clearly see that both terms are very much greater than 1 which is the upper bound on $S(n)$. In computing each of $s_e(n)$ and $s_o(n)$ there has already been some loss of precision which is insignificant for the values of $s_e(n)$ and $s_o(n)$ themselves, but when we take their difference the loss of precision creates a significant impact. As a result we observe the same anomalous behavior in Figure 2 as we did in Figure 1.

n	$s_e(n)$	$s_o(n)$
5	1.579812e+01	1.571134e+01
10	5.161556e+03	5.161535e+03
15	2.020793e+06	2.020793e+06
20	8.776530e+08	8.776530e+08
25	4.060365e+11	4.060365e+11
30	1.959499e+14	1.959499e+14

Table 2: Sums of even and odd terms: Method A2

The lessons to be learned from this section are twofold. First, we should not use an alternating series in which the partial sum is close to the next term in the sequence. Second, we should not rearrange the series such that large numbers that have to be subtracted from each other to estimate (4). In the next section we propose an alternate method to evaluate the RHS of (4) in which each term in the summation is positive, and thus we eliminate the problems of this section.

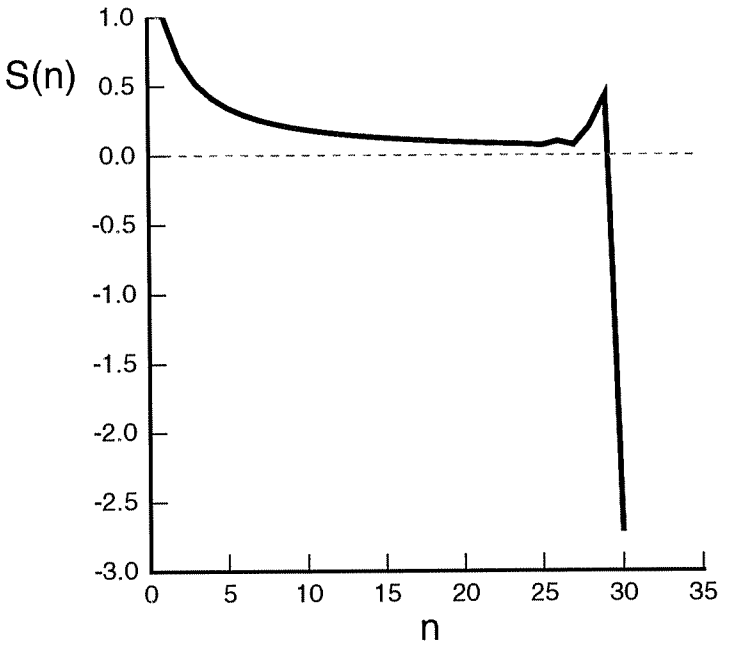

Figure 2: $S(n)$ versus n: Method A2

4 Alternate Method of Summation

In this section we explore an alternate method for summing (4). Rather than summing (4) as an alternating series of positive and negative terms we express it as a sum of positive terms only as given by the following theorem.

Theorem 4.1

$$S(n) = \sum_{i=0}^{\infty} \frac{d_i^{n-1}}{n^{i+1}}. \tag{8}$$

where

$$d_i^n = \sum_{j=0}^{i} d_j^{n-1} \frac{i!}{j!(i+1-j)!}, \quad n > 0, \quad \textit{and} \quad d_i^0 = \begin{cases} 1 & i = 0 \\ 0 & \textit{otherwise} \end{cases}$$

Note that (8) is an infinite series; we will study its numerical properties in Section 5 where we observe how quickly the series converges to its limit.

Proof of Theorem 4.1:

To prove this theorem we start out with the expression for $S(n)$ as given by equation (4). That is,

$$S(n) = \frac{1}{(n-2)!} \sum_{i=0}^{n-2} \binom{n-1}{i} (-1)^i (n-i)^{n-2} \ln(n-i).$$

In order to obtain a series of positive terms only we need to remove the $(-1)^i$ from the series. However, it is not possible to simplify the series directly because of the presence of the $\ln(n-i)$ term. As a result we first write $\ln(n-i)$ as $\ln(n(1-i/n)) = \ln n + \ln(1-i/n)$, where $\ln(1-i/n)$ will be rewritten using a power series expansion. We have,

$$\begin{aligned}
S(n) &= \frac{1}{(n-2)!} \sum_{i=0}^{n-2} \binom{n-1}{i} (-1)^i (n-i)^{n-2} \left\{ \ln n + \ln\left(1 - \frac{i}{n}\right) \right\} \\
&= \frac{1}{(n-2)!} \ln n \sum_{i=0}^{n-2} \binom{n-1}{i} (-1)^i (n-i)^{n-2} \;+\; \frac{1}{(n-2)!} \sum_{i=0}^{n-2} \binom{n-1}{i} (-1)^i (n-i)^{n-2} \ln\left(1 - \frac{i}{n}\right) \\
&= 0 + \frac{n^{n-2}}{(n-2)!} \sum_{i=0}^{n-2} \binom{n-1}{i} (-1)^i \left(1 - \frac{i}{n}\right)^{n-2} \ln\left(1 - \frac{i}{n}\right),
\end{aligned}$$

where the first summation is zero by virtue of Proposition 2.1 (set $m = n-1$, $a = 0$, $b = 1$, $c = n$, and $\ell = n-2 < n-1$). Let $p(x) = -(1-x)^{n-2} \ln(1-x)$ so that

$$S(n) = \frac{n^{n-2}}{(n-2)!} \sum_{i=0}^{n-2} \binom{n-1}{i} (-1)^{i+1} p(i/n).$$

Taking the power series expansion of $p(x)$ we get

$$p(x) = -(1-x)^{n-2} \ln(1-x) \;=\; \sum_{j=1}^{\infty} a_j x^j.$$

Note that $a_0 = 0$ because $p(0) = 1 . \ln 1 = 0$. We will determine the coefficients a_j once we simplify $S(n)$ a bit further. We now have

$$\begin{aligned}
S(n) &= \frac{n^{n-2}}{(n-2)!} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^{i+1} \sum_{j=1}^{\infty} a_j \frac{i^j}{n^j} \\
&= \frac{n^{n-2}}{(n-2)!} \sum_{j=1}^{\infty} \frac{a_j}{n^j} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^{i+1} i^j.
\end{aligned}$$

By Proposition 2.1 the inner sum evaluates to zero for $j < n-1$. Therefore,

$$\begin{aligned} S(n) &= \frac{n^{n-2}}{(n-2)!} \sum_{j=n-1}^{\infty} \frac{a_j}{n^j} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^{i+1} i^j \\ &= \frac{n^{n-2}}{(n-2)!} \sum_{\ell=0}^{\infty} \frac{a_{n-1+\ell}}{n^{n-1+\ell}} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^{i+1}\, i^{n-1+\ell} \end{aligned} \tag{9}$$

The last equation was obtained by substituting $\ell = j-n-1$, $j \geq n-1$. We now determine the a_j's for $j \geq n-1$. Expanding each of the terms of $p(x)$ as a power series, we get

$$\begin{aligned} p(x) &= -(1-x)^{n-2} \ln(1-x) \\ &= \sum_{j=0}^{n-2} \binom{n-2}{j} (-1)^j x^j \sum_{k=1}^{\infty} \frac{x^k}{k} \\ &= \sum_{k=1}^{\infty} \sum_{j=0}^{n-2} \binom{n-2}{j} (-1)^j \frac{x^{j+k}}{k} \\ &= \sum_{m=1}^{\infty} x^m \sum_{j=0}^{\min(n-2,m-1)} \binom{n-2}{j} (-1)^j \frac{1}{m-j}, \end{aligned}$$

where we substituted $m = j+k$. Since $k \geq 1$, $j \leq m-1$ which means that $j \leq \min(n-2, m-1)$. Comparing this expansion of $p(x)$ with $p(x) = \sum_{m=1}^{\infty} x^m a_m$, we have

$$a_m = \sum_{j=0}^{\min(n-2,m-1)} \binom{n-2}{j} (-1)^j \frac{1}{m-j}, \quad m = 1, 2, \ldots.$$

Since we are only interested in obtaining a_m for $m \geq n-1$, we have

$$\begin{aligned} a_{n-1+\ell} &= \sum_{j=0}^{n-2} \binom{n-2}{j} (-1)^j \frac{1}{n-1+\ell-j}, \qquad \ell \geq 0 \\ &= \sum_{j=0}^{n-2} \binom{n-2}{j} (-1)^j \int_0^1 x^{n+\ell-j-2}\, dx \\ &= \int_0^1 x^{\ell} \sum_{j=0}^{n-2} \binom{n-2}{j} (-1)^j x^{n-2-j}\, dx \end{aligned}$$

$$= \int_0^1 x^\ell (x-1)^{n-2}\, dx$$

$$= (-1)^{n-2} \int_0^1 x^\ell (1-x)^{n-2}\, dx$$

$$= (-1)^{n-2} \frac{\ell!\,(n-2)!}{(n+\ell-1)!}.$$

The last equality holds because

$$\beta(i,j) = \int_0^1 x^{i-1}(1-x)^{j-1} dx \;=\; \frac{\Gamma(i)\Gamma(j)}{\Gamma(i+j)} = \frac{(i-1)!(j-1)!}{(i+j-1)!}.$$

$\beta(i,j)$ is called the beta function. (See [2].) Substituting the above expression for $a_{n-1+\ell}$ in (9) we get

$$S(n) = \frac{n^{n-2}}{(n-2)!} \sum_{\ell=0}^{\infty} (-1)^{n-2} \frac{\ell!\,(n-2)!}{(n+\ell-1)!} \cdot \frac{1}{n^{n-1+\ell}} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^{i+1}\, i^{n-1+\ell}$$

$$= \sum_{\ell=0}^{\infty} (-1)^{n-2} \frac{\ell!}{(n+\ell-1)!} \cdot \frac{1}{n^{\ell+1}} \sum_{i=0}^{n-1} \binom{n-1}{i} (-1)^{i+1}\, i^{n-1+\ell}$$

$$= \sum_{\ell=0}^{\infty} \frac{\ell!}{(n+\ell-1)!} \cdot \frac{1}{n^{\ell+1}} \sum_{i=0}^{\infty} \binom{n-1}{i} (-1)^{n-1+i}\, i^{n-1+\ell}$$

$$= \sum_{\ell=0}^{\infty} \frac{\ell!}{(n+\ell-1)!} \cdot \frac{1}{n^{\ell+1}} \sum_{i=0}^{\infty} \binom{n-1}{i} (-1)^{n-1-i}\, i^{n-1+\ell}$$

$$= \sum_{\ell=0}^{\infty} \frac{\ell!}{(n+\ell-1)!} \cdot \frac{1}{n^{\ell+1}} \sum_{i=0}^{\infty} \binom{n-1}{i} (-1)^{n-1-i} \left[\frac{d^{n-1+\ell}}{dx^{n-1+\ell}} e^{ix} \right]_{x=0}$$

$$= \sum_{\ell=0}^{\infty} \frac{\ell!}{(n+\ell-1)!} \cdot \frac{1}{n^{\ell+1}} \frac{d^{n-1+\ell}}{dx^{n-1+\ell}} \left[\sum_{i=0}^{\infty} \binom{n-1}{i} (-1)^{n-1-i} e^{ix} \right]_{x=0}$$

$$= \sum_{\ell=0}^{\infty} \frac{\ell!}{(n+\ell-1)!} \frac{1}{n^{\ell+1}} \frac{d^{n-1+\ell}}{dx^{n-1+\ell}} \left[(e^x - 1)^{n-1} \right]_{x=0}.$$

The power series expansion of $(e^x-1)^{n-1}$ yields

$$(e^x - 1)^{n-1} = \sum_{k=0}^{\infty} C_k^{n-1} x^{k+n-1},$$

where $C_k^{n-1} \geq 0$. As a result,

$$\frac{d^{n-1+\ell}}{dx^{n-1+\ell}} \left[(e^x - 1)^{n-1}\right]_{x=0} = (\ell + n - 1)!\, C_\ell^{n-1}.$$

Hence

$$\begin{aligned} S(n) &= \sum_{\ell=0}^{\infty} \frac{\ell!}{(n+\ell-1)!} \cdot \frac{1}{n^{\ell+1}} \cdot (\ell + n - 1)!\, C_\ell^{n-1} \\ &= \sum_{\ell=0}^{\infty} \frac{\ell!}{n^{\ell+1}} C_\ell^{n-1} \end{aligned} \tag{10}$$

It now remains to determine C_ℓ^{n-1}. Since

$$(e^x - 1)^{n-1} = \sum_{k=0}^{\infty} x^{k+n-1} C_k^{n-1},$$

it readily follows that (by setting $n = 1$ and $n = 2$)

$$C_k^0 = \begin{cases} 1 & k = 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad C_k^1 = \frac{1}{(k+1)!}. \tag{11}$$

In general we have

$$\begin{aligned} (e^x - 1)^n &= (e^x - 1)(e^x - 1)^{n-1} \\ &= \sum_{\ell=0}^{\infty} \frac{x^{\ell+1}}{(\ell+1)!} \sum_{m=0}^{\infty} x^{m+n-1} C_m^{n-1} \\ &= \sum_{\ell=0}^{\infty} \sum_{m=0}^{\infty} x^{m+\ell+n} \frac{C_m^{n-1}}{(\ell+1)!} \\ &= \sum_{k=0}^{\infty} x^{k+n} \sum_{m=0}^{k} \frac{C_m^{n-1}}{(k+1-m)!} \end{aligned}$$

where we substituted $k = m + \ell$. Comparing this form for $(e^x - 1)^n$ with

$$(e^x - 1)^n = \sum_{k=0}^{\infty} x^{k+n} C_k^n$$

we obtain

$$C_k^n = \sum_{m=0}^{k} \frac{C_m^{n-1}}{(k+1-m)!}. \tag{12}$$

This recursive formula for C_k^n thus enables us to compute $S(n)$ as a sum of nonnegative terms given by (10). A somewhat simpler form of $S(n)$ is obtained from equation (10) as

$$\begin{aligned} S(n) &= \sum_{\ell=0}^{\infty} \frac{\ell!}{n^{\ell+1}} C_\ell^{n-1} \\ &= \sum_{\ell=0}^{\infty} \frac{d_\ell^{n-1}}{n^{\ell+1}}, \end{aligned}$$

where

$$d_i^n = i!\, C_i^n = i! \sum_{m=0}^{i} \frac{C_m^{n-1}}{(i+1-m)!} = \sum_{m=0}^{i} m!\, C_m^{n-1} \frac{i!}{m!\,(i+1-m)!} = \sum_{m=0}^{i} d_m^{n-1} \frac{i!}{m!(i+1-m)!}, \quad n > 0, \tag{13}$$

and

$$d_i^0 = C_i^0 = \begin{cases} 1 & i = 0 \\ 0 & \text{otherwise.} \end{cases}$$

■

5 Numerical Evaluation of the Infinite Series

The infinite series (8) can be used to provide accurate values for $S(n)$ provided that good error estimates are available for the truncated series. The next lemma provides estimates for the coefficients d_i^n.

Lemma 5.1

$$d_i^n \le \frac{i!}{(n+i)!} n^{n+i}, \quad n \ge 1, \quad i \ge 0. \tag{14}$$

Proof. By definition $d_i^n = i!\, C_i^n$ (see (13)); so we need to show that

$$C_i^n \le \frac{n^{n+i}}{(n+i)!}, \quad n \ge 1, \quad i \ge 0. \tag{15}$$

We prove this by induction on n. From (11) we note that $C_i^1 = 1/(i+1)!$ which shows that (15) is true for $n = 1$, $i \geq 0$. Assume that (15) holds for $n = 1, 2, \ldots, k-1$. We show that it also holds for $n = k$. We have

$$\begin{aligned}
C_i^k &= \sum_{m=0}^{i} \frac{C_m^{k-1}}{(i+1-m)!} \qquad \text{from (12)} \\
&\leq \sum_{m=0}^{i} \frac{(k-1)^{k-1+m}}{(k-1+m)!\,(i+1-m)!} \qquad \text{induction hypothesis} \\
&= \frac{1}{(k+i)!} \sum_{m=0}^{i} \binom{k+i}{k+m-1} (k-1)^{k+m-1} \\
&= \frac{1}{(k+i)!} \sum_{j=k-1}^{k+i-1} \binom{k+i}{j} (k-1)^j \\
&\leq \frac{1}{(k+i)!} \sum_{j=0}^{k+i} \binom{k+i}{j} (k-1)^j \\
&= \frac{(k-1+1)^{k+i}}{(k+i)!} \\
&= \frac{k^{k+i}}{(k+i)!},
\end{aligned}$$

which completes the proof by induction. ∎

Using Lemma 5.1 we now estimate the remainder of the series (8) defined by

$$R(L,n) \equiv \sum_{i=L}^{\infty} \frac{d_i^{n-1}}{n^{i+1}}.$$

Theorem 5.1

$$R(L,n) \leq U(L,n) \equiv \frac{L!}{(n+L-1)!}(n-1)^{n-1}(1-1/n)^L. \tag{16}$$

Proof.

$$\begin{aligned}
R(L,n) &= \sum_{i=L}^{\infty} \frac{d_i^{n-1}}{n^{i+1}} \\
&\leq \sum_{i=L}^{\infty} \frac{i!\,(n-1)^{n-1+i}}{n^{i+1}\,(n-1+i)!} \qquad \text{Lemma 5.1}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=L}^{\infty} \frac{i!}{(n-1+i)!} \left(\frac{n-1}{n}\right)^{n-1+i} n^{n-2}, \quad n \geq 2 \\
&= n^{n-2} \sum_{i=L}^{\infty} \frac{1}{(n-1+i)\cdots(i+1)} (1-1/n)^{n-1+i} \\
&\leq n^{n-2} \frac{1}{(n-1+L)\cdots(L+1)} \sum_{i=L}^{\infty} (1-1/n)^{n-1+i} \\
&= \frac{L!}{(n+L-1)!} n^{n-2} (1-1/n)^{n+L-1} \sum_{j=0}^{\infty} (1-1/n)^{j} \\
&= \frac{L!}{(n+L-1)!} n^{n-2} (1-1/n)^{n+L-1} \frac{1}{1-(1-1/n)} \\
&= \frac{L!}{(n+L-1)!} n^{n-1} (1-1/n)^{n-1} (1-1/n)^{L},
\end{aligned}
$$

which simplifies to (16). ∎

Theorem 5.1 provides us with an upper bound on the number of terms needed for the truncated series to achieve any desired accuracy ϵ. Thus summing (8) up to M terms guarantees an absolute error of less than ϵ in $S(n)$, where $M = \min\{L-1 \geq 0 : U(L,n) \leq \epsilon\}$. Since $U(L,n)$ in (16) contains factorials and high powers of n we can alternatively check if $g(L,n) \equiv \ln(U(L,n)) \leq \ln(\epsilon)$. We need not compute $g(L,n)$ separately for each value of L since we can recursively express $g(L,n)$ in terms of $g(L-1,n)$ as shown below. This enables us to check for termination while computing the partial sums of (8). To establish the recursion in $g(L,n)$ we proceed as follows:

$$
\begin{aligned}
U(L,n) &= \frac{L!}{(n+L-1)!} (n-1)^{n-1} (1-1/n)^{L} \\
U(L+1,n) &= \frac{(L+1)!}{(n+L)!} (n-1)^{n-1} (1-1/n)^{L+1} \\
&= \frac{L+1}{n+L} (1-1/n) U(L,n), \quad L \geq 1.
\end{aligned}
$$

$$
g(L+1,n) \equiv \ln(U(L+1,n)) = \ln\left(\frac{L+1}{n+L}\right) + \ln(1-1/n) + \ln(U(L,n))
$$

$$= \quad g(L,n) - \ln\left(\frac{n+L}{L+1}\right) - \ln\left(\frac{n}{n-1}\right), \quad L \geq 1,$$

where

$$g(1,n) = \ln(U(1,n)) = -\ln(n!) + (n-1)\ln(n-1) + \ln(n-1) - \ln(n) = -\sum_{i=2}^{n} \ln(i) + n \cdot \ln(n-1) - \ln(n).$$

Algorithm 1 presents pseudo-code for estimating (8) within an absolute accuracy of ϵ. The algorithm starts out with $L = 1$ and stops at the value of L for which $U(L,n)$ (denoted by remainder) is less than or equal to a prespecified ϵ. To compute d_L^{n-1} it is necessary to have obtained all d_j^{n-2}, $j = 0, \ldots, L$, as seen from (13). The algorithm computes d_L^i, $i = 1, 2, \ldots, n-1$ at each step L and thus when it is time to compute d_L^{n-1}, the previous d_j^{n-2}, $j = 0, \ldots, L$, have already been computed.

ALGORITHM 1

begin
L := 1; Sn := 1/n; product := n;
remainder := $n * \ln(n-1) - \ln(n) - \sum_{i=2}^{n} \ln(i)$;
tn := $\ln(n/(n-1))$; { temporary variable }
eps := $\ln(\epsilon)$;
while (remainder > eps) **do**
begin
    d_L^1 := 1/(L+1);
    for i := 2 to $n-1$ **do**
    begin
        d_0^{i-1} := 1;
        sum := d_L^{i-1};
        for j := $L-1$ downto 0 **do** { compute d_L^i using nested multiplication }
            sum := sum * $(L+1-j)/(j+1) + d_j^{i-1}$;
        d_L^i := sum/(L+1);
    end;{for}
    product := product * n;
    Sn := Sn + d_L^{n-1}/product;
    L := L + 1;
    remainder := remainder − ln((n + L)/(L + 1)) − tn;
end;{while}
M := $L-1$;
writeln('Sn = ',Sn,' number of terms = ',M);
end.

Table 3 presents estimates of $S(n)$, $n = 20, 40, 60, 80, 100$, along with the number of terms, M, in the truncated series, as computed by Algorithm 1. We observe from Table 3 that M increases with n implying that as we increase n a larger number of terms are needed to obtain an absolute error of less than ϵ. This is because the expression for $U(L, n)$ in (16) contains a high power of n, i.e., $(n-1)^{n-1}$. For example, consider the first error estimate $U(1, n) = 1/n!\ (n-1)^{(n-1)}(1 - 1/n)$. For n large, we can ignore the $(1 - 1/n)$ term and use Stirling's approximation $(n-1)! \approx \sqrt{2(n-1)\pi}(n-1)^{n-1}/e^{n-1}$, to get $U(1, n) \approx e^{n-1}/(n\sqrt{2(n-1)\pi}) \gg 1$. However, we know that $S(n) \leq 1$ by the problem definition and thus $R(L, n) \leq 1$, which shows that $U(1, n)$ is an extremely pessimistic bound for $R(1, n)$, when n is large. The factor $(n-1)^{n-1}$ also dominates the error estimates at $L = 2, 3$, and so on until we reach a sufficiently large L so that the $(n+L-1)!$ term in the denominator of $U(L, n)$ reduces its impact. The extremely pessimistic estimates at low values of L causes the overall estimate of M to be rather pessimistic. We observe from Table 3 that the estimates of $S(n)$ are the same for $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$ at each value of n, even though the values of M are not the same for these two ϵ's. Hence for $\epsilon = 10^{-6}$ the truncated series has already converged much before the estimated number of terms for this value of ϵ. Nevertheless, from a computation point of view we note that even for $n = 100$ and $\epsilon = 10^{-6}$ the time required by Algorithm 1 was less than 1 second on a DECstation 3100 (the code was written in C and used double precision arithmetic) and thus the pessimistic estimates of M did not pose computational limitations.

n	ϵ	$S(n)$	M
20	10^{-3}	0.096667	19
	10^{-6}	0.096667	31
40	10^{-3}	0.049167	33
	10^{-6}	0.049167	39
60	10^{-3}	0.032963	41
	10^{-6}	0.032963	49
80	10^{-3}	0.024792	52
	10^{-6}	0.024792	60
100	10^{-3}	0.019867	63
	10^{-6}	0.019867	71

Table 3: Estimates for $S(n)$,$n = 20, 40, 60, 80, 100$

As noted above the upper bound on the remainder seems quite pessimistic since the error estimates at low values of L are unbelievably large. We now plot the partial sums $S(L,n)$ versus L in Figure 3 to test the *actual* convergence rate of the partial series as opposed to the estimates provided above. The partial sum $S(L,n)$ is defined by

$$S(L,n) \equiv \sum_{i=0}^{L} \frac{d_i^{n-1}}{n^{i+1}}.$$

From Figure 3 we note that $S(L,n)$ converges quite rapidly with L, for all values of n in the figure, and within $L = 10 - 15$ it reaches its limiting value. This suggests that a tighter bound on $R(L,n)$ exists than the one in (16), specially for low values of L. Although we have obtained a tighter bound on d_L^n for low L it yielded only a weak bound on $R(L,n)$, which was useful only for high values of ϵ like $0.1 - 0.01$ and it resulted in the same bound as (16) for more typical values of ϵ like $10^{-4} - 10^{-6}$.

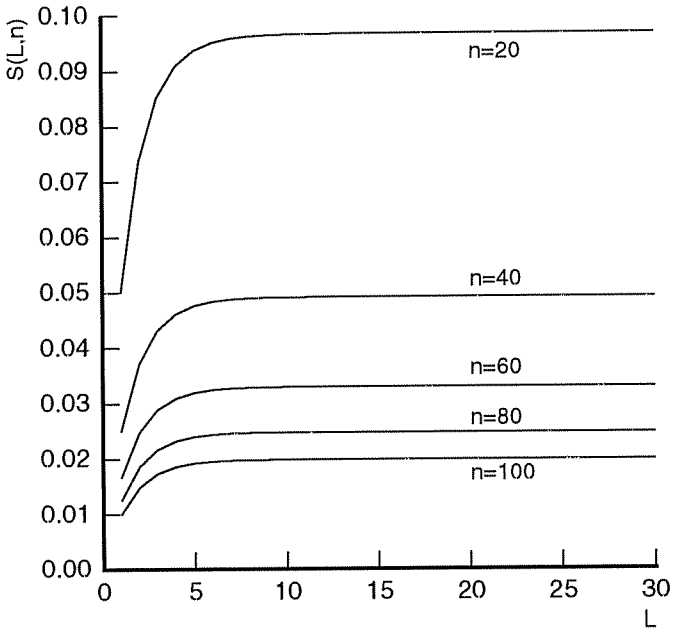

Figure 3: $S(L,n)$ versus L

6 Mean Barrier Completion Times of the Model

Using the results of Section 5 we computed the mean barrier completion time $S(n)$ versus n for $n = 3$ to $n = 100$ (the curve is almost flat for higher values of n and for $n = 1$ and $n = 2$ we simply use $S(1) = 1$

and $S(2) = \ln 2$ as seen from equation (4)). We evaluated the partial sums to an accuracy of $\epsilon = 10^{-6}$. The value of 10^{-6} for ϵ is reasonable within the range of n from 2 to 100 since we know that $S(n) \geq 1/n$ (equal task service times result in $S(n) = 1/n$ and the unequal task times given by (1) result in a higher value for $S(n)$), which means that $S(n) \geq 0.01$ for $n = 100$ resulting in a relative error less than or equal to $10^{-6}/10^{-2} = 10^{-4}$ or 0.01% for all n between 1 and 100. We compare these values of $S(n)$, $n = 1, 2, \ldots, 100$ against mean barrier completion times for two other program models below.

Figure 4 plots mean barrier completion time on n processors versus n for $n = 1, 2, \ldots, 100$; the curve for the model analyzed in this paper is labelled as "uniform". The curve labelled "linear" corresponds to mean barrier completion times for equal task service times ($T_i = D/n$, $i = 1, 2, \ldots, n$) or linear speedups (i.e., $S(n) = 1/n$, $i = 1, 2, \ldots, n$, assuming $D = 1$). The curve labelled "exponential" corresponds to mean barrier completion times for independent and identically distributed (i.i.d.) task service times from an exponential distribution ($T_i = \exp(\mu)$, $i = 1, 2, \ldots, n$), a model that has been frequently used in the literature (for example in [7, 6, 4, 8]). For i.i.d. exponential task times $T_i = \exp(\mu)$, $i = 1, 2, \ldots, n$, we have (cf. [7, 6])

$$S(n) = E[\max(T_1, \ldots, T_n)] \;=\; n/\mu + (n-1)/\mu + \ldots + 1/\mu \;=\; H_n/\mu, \tag{17}$$

where H_n is the harmonic sum $\sum_{i=1}^{n} 1/i$. The interpretation of the above equation is that time to first task completion is the mean of the minimum of n exponentials which equals n/μ, and then the recursion continues with the remaining $n-1$ tasks and so on, by virtue of the memoryless property of the exponential distribution. In Figure 4, for the exponential model we have divided $S(n) = H_n/\mu$ by the mean job demand n/μ to compare it on the same scale as the linear and uniform models of parallel programs (for which demand, D, is set to 1).

We see from Figure 4 that the model analyzed in this paper results in mean completion times in between those of the exponential task service times model and those of the equal task service times model. Thus, the estimates obtained for the uniform model in this paper are less pessimistic than the estimates for the exponential task service times model. Interestingly, the two simulation studies of multiprogrammed parallel processor allocation policies by Majumdar et al. [5] and Leutenegger [3] mentioned that their *qualitative results* of policy comparison were more or less the same for the uniform model and the equal task service times model.

Besides yielding more optimistic estimates than the exponential model the uniform model also has the desirable feature of allowing one to test the impact of job demand characteristics on mean response time.

From (1) we note that each task service time is derived from the total demand D and thus various distributions of total job demand can be used to study the sensitivity of scheduling policies to higher moments of demand than just the first moment (for example, the variance of demand is an important characteristic that determines the performance of uniprocessor and multiprocessor policies). On the other hand such a sensitivity study is not possible from the exponential model because the distribution of job demand is fixed for a given number of tasks (since total job demand for n tasks is a sum of n exponentially distributed random variables, or an n stage Erlang distribution).

This paper also sheds light on why the exponential model has been used so often in the past even though the exponential model does not provide much flexibility in terms of studying the impact of demand characteristics on response time. As seen from equation (17) obtaining $S(n)$ for the exponential model is extremely simple and there is also no numerical problem in computing it. On the other hand the uniform model, which appeared simple to state, turned out to be complex to analyze and even to compute a simple estimate like mean barrier completion time, $S(n)$, detailed rearrangements had to be done. This seems to indicate a tradeoff between representation of characteristics of parallel workloads and ease of analysis. Some implications of this observation in terms of future models are:

1. Use approximate analysis instead of exact analysis. We saw in this paper that exact analysis to compute $S(n)$ for the uniform model is rather complex. Obtaining mean response time estimates for a parallel workload will be considerably more complex than estimating mean service time estimates like $S(n)$ and even a closed form solution may not exist for the mean response time estimates. This motivates the use of approximate analysis for future models that are as complex to analyze as the one in this paper.

2. If one prefers to do exact analysis for a model like the one in this paper, then one should be careful to ensure that the solution does not become numerically unstable as the number of processors increases. We hope that the techniques developed in this paper will useful in this regard, besides being of interest from a mathematical point of view.

3. Search for simpler models for parallel programs than the uniform model, that retain the essential characteristics of parallel workloads.

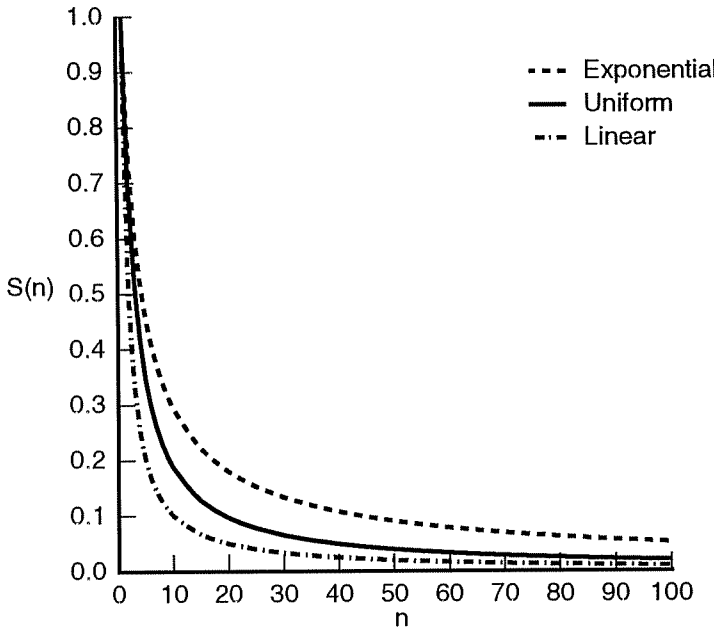

Figure 4: Mean Barrier Completion Times on n processors

Acknowledgements

We are grateful to Professor Mary Vernon for providing useful comments on the material in this paper.

References

[1] V. ADVE, and M. VERNON. Influence of random delays on parallel execution times. *ACM SIGMETRICS Conf. and Performance Evaluation Review 21*, 1 (May 1993).

[2] E. DUDEWICZ, and S. MISHRA. *Modern Mathematical Statistics*. John Wiley & Sons, New York, 1988.

[3] S. LEUTENEGGER. Issues in multiprogrammed multiprocessor scheduling. Ph.D. Thesis, Technical Report #954, Computer Sciences Dept., Univ. of Wisconsin-Madison, August 1990.

[4] S. LEUTENEGGER, and R. NELSON. Analysis of spatial and temporal scheduling policies for semi-static and dynamic multiprocessor environments. Research Report - IBM T.J. Watson Research Center, Yorktown Heights, August 1991.

[5] S. MAJUMDAR, D. EAGER, and R. BUNT. Scheduling in multiprogrammed parallel systems. *ACM SIGMETRICS Conf. and Performance Evaluation Review 16*, 1 (May 1988), 104-113.

[6] R. NELSON. A performance evaluation of a general parallel processing model. *ACM SIGMETRICS Conf. and Performance Evaluation Review 18*, 1 (May 1990), 13-26.

[7] R. NELSON, D. TOWSLEY, and A. TANTAWI. Performance analysis of parallel processing systems. *IEEE Trans. on Software Engg. 14*, 4 (April 1988), 532-540.

[8] R. NELSON, and D. TOWSLEY. A performance evaluation of several priority policies for parallel processing systems. COINS Tech. Report 91-32, Computer and Info. Sciences, Univ. of Mass. at Amherst, May 1991. (To appear in JACM.)