*Supported by the
National Science Foundation
Engineering Research Center
Program (NSFD CD 8803012),
Industry and the University*

Optimal Scheduling for a Distributed Parallel Processing Model

by A.M. Makowski and R. Nelson

TR 92-36

RC 17449 (#77995) 2/21/92
Computer Science 50 pages

Research Report

Optimal Scheduling for a Distributed Parallel Processing Model

Armand M. Makowski

Electrical Engineering Department
and Systems Research Center
University of Maryland
College Park, MD 20742

Randolph Nelson

IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

Optimal Scheduling for a Distributed Parallel Processing Model

Armand M. Makowski
Electrical Engineering Department
and Systems Research Center
University of Maryland
College Park, MD 20742

Randolph Nelson
IBM Research Division
T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

February 19, 1992

Abstract

We consider a model of a parallel processing system consisting of K distributed homogeneous processors each with private memory in which tasks queue before being served. Jobs arriving to the system consist of a set of tasks which can be executed independently of each other and we consider a job to be completed only after all of its component tasks have finished execution. A central dispatcher schedules the tasks on the processors at job arrival instants using information on the number of tasks currently scheduled on each processor. We model this system as a distributed fork/join queueing system and derive the structure of the individually optimal scheduling policy. Our results show that the individually optimal policy is a mixture of policies corresponding to sequential job execution (all tasks are scheduled on a single processor) and parallel scheduling (tasks are distributed among several processors in a manner that tends to equalize queue lengths). We show that, under conditions that include the case of moderate to heavy loads, the individually optimal scheduler schedules tasks according to the sequential policy which runs counter to the intuition that parallel processing is desirable. Because we do not include certain overheads associated with executing jobs in parallel in our model, our results are biased towards parallel rather than sequential processing. Thus our results strongly suggest that for actual distributed memory systems the benefits of parallel processing can be achieved only in conditions of light load. Response time properties of the individually optimal scheduler are derived and compared by simulation to other scheduling policies.

1 Introduction

In this paper we consider a simple model of a parallel processing system consisting of multiple distributed processors that service jobs consisting of a number of independent tasks. Tasks can be executed on any processor and we assume that a job is completed when all of its tasks have finished executing, a time which we term a *synchronization* point. These assumptions on the workload are strongly biased in favor of parallel processing since we assume that there is no overhead associated with parallel execution and that no data dependencies exist between tasks. Although the model ignores these system overheads, it does include the effects of random service times. For example, even if all tasks begin executing simultaneously, variations in service time imply that the finishing times will be staggered and thus that some tasks must wait for their siblings to finish executing before synchronization.

Our objective in the paper is to determine the "best" way to schedule the tasks composing a job. There are two scheduling disciplines that motivate our investigations. In the fully parallel policy all tasks composing a job are spread "evenly" among the processors of the system, thus attempting to achieve the maximum possible parallel job execution. Clearly, for such a scheduling discipline the time needed to reach a synchronization point is also maximized. The second scheduler is fully sequential and avoids synchronization altogether by scheduling all tasks composing a job on a single processor. In essence, these two policies correspond to the extremes of scheduling independent tasks; the optimal scheduler we derive bridges the gap between them and reveals in greater detail the subtleties of the trade–off between fully parallel and fully sequential execution. An indication of this trade–off is found in the fact, as we later show, that the optimal scheduler converges to an extreme policy under appropriate load conditions; the optimal policy is fully parallel in light load and fully sequential in heavy load. The scheduling policy depends upon the distribution of task service demand and generally favors sequential execution as the variance of service time increases. A principle result of our analysis is that , for all distributions with non–zero variance, the fully parallel policy is a poor choice in heavy load conditions. Since we expect future distributed parallel systems to be often run under conditions of heavy load, this suggests that such systems might be better served by sequential scheduling algorithms.

The above conclusions are even more striking considering that our model tends to favor parallel execution since it ignores system overheads that are found in actual systems. Such costs range from contention for shared resources such as memory access and interconnection bandwidth, to overheads associated with synchronizing the computation among the tasks [11, 18, 33]. Synchronization costs include the time and resources used in the synchronization protocol to establish when all tasks have finished execution as well as inherent randomness associated with task execution times. Previous work on these issues has been carried out in both deterministic and probabilistic frameworks. Deterministic models that assume unbounded parallelism and include effects of the synchronization protocols are discussed in [1, 10, 11, 17, 18]; models that ignore these costs and analyze the effects that randomness in task execution time have on synchronization are considered in [2, 3, 4, 5, 34, 35]. Work relating the effects of scheduling policies on response times on a parallel processing environment that include synchronization overheads can be found in [24, 27, 38, 41].

In our model we assume that once a task is scheduled on a given processor it will be executed on that processor. That is, we concern ourselves with *static* scheduling policies, in contrast to *dynamic* policies which allow tasks to migrate between processors as a function of the state of the system. We point the reader to references [13, 14, 15, 22, 23, 26, 30, 31, 40] where migration policies for sequential jobs are analyzed under various system architectures. Our model is applicable to situations where such migrations are not allowed. Such a restriction might be imposed on the scheduler to limit scheduling complexity or because the benefits of a dynamic policy are determined to be small with respect to the costs associated with migrating tasks. The results we present here can, we believe, be used to guide the construction of dynamic policies for distributed systems. For example, it might be reasonable to initially schedule tasks in the system according to the static policy under the assumption that there is a large probability that no migrations take place before a scheduled job completes execution. Examples of applications of our model are that of a set of independent processors that communicate over a local area network. An example of such a system is the Condor system at the University of Wisconsin which provides facilities for distributed processing over a set of workstations [25]. Another situation that can be modeled can be found in computer systems with non–uniform memory access (NUMA) where the cost to access nonlocal memory is very expensive.

In [19, 35] a similar model was analyzed under the assumption that all tasks of a job were evenly distributed among the processors of the system while in [36] all tasks were assumed scheduled on the processor that had a minimal number of tasks in its local queue (these polices correspond to the two extreme policies discussed above). Here we identify the policy which minimizes the expected execution time of each *individual* job. This policy, termed the *individually* optimal policy, differs from a *socially* optimal policy which attempts to minimize the expected job response time for all job arrivals to the system. The individually optimal policy can be derived in first–come first–serve systems by not accounting for the effects that scheduling decisions have on future arrivals. Although future arrivals do influence the expected response time averaged over all arrivals, we believe that the policy derived here is sufficiently close to the socially optimal policy to be of practical use. Perhaps more importantly the analysis reveals some interesting tradeoffs found in parallel processing scheduling, some of which contradict the intuition that parallel processing is always beneficial.

The paper is organized as follows: Section 2 presents the mathematical model that we use to establish properties of the optimal scheduler. Section 3 first reviews the properties of the optimal policy for the case $K = 2$. This case was analyzed in a companion paper [28] and is used as a touchstone for developments of the more complicated general case. Section 3 then develops some auxiliary results that are used to obtain the properties of the optimal schedule in the case of an arbitrary number of processors. In section 4 we present comparison results on the expected response time for the optimal policy, and we compare them to the two extreme policies mentioned above, namely the one which schedules all the queues evenly and the one which only schedules the shortest queue. In section 5 we present our conclusions and suggest future research. The appendices contain several technical results which are not essential for understanding the main conclusions of the paper.

A few words on the notation used in this paper: We denote the set of non–negative integers by $\mathbb{N}$, and the set of all real numbers by $\mathbb{R}$. Throughout the paper, K always denotes a given positive integer. For every positive integer L, the l^{th} component of any element $\mathbf{x}$ in $\mathbb{R}^L$ is denoted by x_l, $l = 1, \ldots, L$, so that $\mathbf{x} \equiv (x_1, \ldots, x_L)$. Moreover, we denote the cardinality of a set E by $|E|$.

2 The Model and The Scheduling Problem

In this paper, we consider a queueing model, depicted in figure 1, consisting of $K \geq 2$ homogeneous single server queues. Each queue has infinite capacity. Task service times at the queues are independent and identically distributed (i.i.d.) non–negative random variables (r.v.'s). We let X be the r.v. denoting the generic service time at any one of the queues and we denote its expected value by $\overline{X}$ and its variance by σ_X^2. Each incoming job is assumed to consist of a random number of tasks described by an integer–valued r.v. B. A job is considered to be finished when all its tasks have finished execution. This corresponds to a synchronization event in the underlying parallel processing model.

We denote by n_k the number of tasks in queue $k, k = 1, 2, \ldots, K$, at a job arrival instant and we write $\mathbf{n} \equiv (n_1, n_2, \ldots, n_K)$ for the vector state of the system (which is an element of $I\!N^K$). Upon job arrival, the value b of B is declared and a dispatcher schedules the job's tasks on the queues using the values of $\mathbf{n}$ and B. We are interested in determining the scheduling policy which, for each arriving job, schedules tasks so that the expected response time of the incoming job is minimized. This policy is the *individually* optimal policy for the system, and as noted in the introduction, differs from the socially optimal policy which minimizes the expected response time over all job arrivals. To avoid cumbersome language, henceforth when we say optimal policy, we mean individually optimal policy.

To define the optimal scheduling problem considered in this paper, let $\{X_j^k, j = 1, 2, \ldots, k = 1, 2, \ldots, K\}$ be i.i.d. non–negative r.v.'s distributed as X and set

$$S_n^k \equiv \sum_{j=1}^{n} X_j^k, \quad k = 1, 2, \ldots, K, \quad n = 1, 2, \ldots \tag{1}$$

with the convention $S_0^k = 0$. Note that $ES_n^k = n\overline{X}$. The value of S_n^k is assumed to be the time needed to execute all the jobs scheduled on processor k given that its queue contains n tasks. This assumption implies that scheduling decisions are made using information that can be summarized by the number of tasks already on processor queues. In particular, we assume either that tasks in the servers have just started execution upon job arrival or that the remaining service time (i.e., residual life) has an identical distribution to that of a service time (in which case the service times are necessarily exponentially distributed).

An admissible scheduling policy is a mapping

$$\pi : \mathbb{N} \times \mathbb{N}^K \to \mathbb{N}^K : (b, \mathbf{n}) \to \pi(b; \mathbf{n}) \equiv (\pi_1(b; \mathbf{n}), \ldots, \pi_K(b; \mathbf{n})) \tag{2}$$

satisfying

$$\sum_{k=1}^{K} \pi_k(b; \mathbf{n}) = b, \quad b = 1, 2, \ldots, \quad \mathbf{n} \in \mathbb{N}^K. \tag{3}$$

In words, when the system is in state $\mathbf{n}$ and the incoming job is made up of b tasks, policy π schedules $\pi_k(b; \mathbf{n})$ of the incoming tasks on processor k , $k = 1, 2, \ldots, K$. The resulting response time of this incoming job is given by

$$T_\pi(b; \mathbf{n}) = \max_{k : \pi_k(b; \mathbf{n}) > 0} \left\{ S^k_{n_k + \pi_k(b; \mathbf{n})} \right\}, \quad b = 1, 2, \ldots, \quad \mathbf{n} \in \mathbb{N}^K. \tag{4}$$

We let

$$\overline{T}_\pi(b; \mathbf{n}) \equiv E T_\pi(b; \mathbf{n}), \quad b = 1, 2, \ldots, \quad \mathbf{n} \in \mathbb{N}^K \tag{5}$$

be the expected response time of the incoming job of size b when the state of the system at time of arrival is $\mathbf{n}$ and policy π is used.

The collection of all admissible scheduling policies is denoted by $\mathcal{P}$. We seek to determine the scheduling policies in $\mathcal{P}$ which minimize the expected response time (5). We denote any such optimal policy by Ψ which is characterized by

$$\overline{T}_\Psi(b; \mathbf{n}) \leq \overline{T}_\pi(b; \mathbf{n}), \quad \pi \in \mathcal{P}, \quad b = 1, 2, \ldots, \quad \mathbf{n} \in \mathbb{N}^K. \tag{6}$$

In discussing the properties of the optimal schedule Ψ, it is clear that we can relabel the queues without loss of generality in our results. This is a direct consequence of the underlying statistical assumptions and of the symmetry property of the maximum operator. To simplify the notation at various places in the exposition, it will be convenient to sometimes assume a labeling of the queues so that $n_1 \leq n_2 \leq \ldots \leq n_K$.

3 K = 2 – The Optimal Policy

In this section we review the structure of the optimal policy for the case where $K = 2$; these results were derived in [28] to which the reader is referred for additional details: We assume that the state upon the arrival of a tagged job is given by $\mathbf{n} = (n_1, n_2)$ and that there are $b \geq 1$ tasks to be scheduled. As pointed out earlier, it is convenient to assume, with no loss of generality, a labeling of the queues so that $n_1 \leq n_2$. Thus we can consider queue 1 (resp. 2) to be the shortest (resp. longest) queue (with the convention that queue 1 is shortest if $n_1 = n_2$). We also define $\Delta\mathbf{n} \equiv n_2 - n_1$.

The optimal policy can be described as operating in two steps. First the policy must determine whether one or both queues are to be scheduled. If only one queue is to be scheduled, then the policy must determine which queue is allocated the b tasks. If both queues are to be scheduled, it then suffices to determine how many tasks are assigned to each queue. These properties are described below.

3.1 Load Balancing Properties of The Optimal Policy

As in [28], we say that if, for the given state $\mathbf{n}$, the policy Ψ schedules the b tasks on more than one queue, then **BQ** (both queues) is satisfied in state $\mathbf{n}$, otherwise if Ψ schedules all b tasks on a single queue, then **SQ** (single queue) is satisfied in state $\mathbf{n}$. The collection of all non–degenerate allocation vectors for assigning b tasks to both queues is the set $\mathcal{B}(b)$ given by

$$\mathcal{B}(b) \equiv \{\mathbf{b} = (b_1, b_2) \in I\!N^2 : 0 < b_1, b_2 < b; \;\; b_1 + b_2 = b\}. \tag{7}$$

From these definitions, we see that **BQ** (resp. **SQ**) is satisfied in state $\mathbf{n}$ when scheduling b tasks provided the condition

$$\min_{\mathbf{b}\in\mathcal{B}(b)} E \max_{k=1,2}\{S^k_{n_k+b_k}\} \leq (\text{resp. } \geq) \min_{k=1,2} ES^k_{n_k+b} \tag{8}$$

holds with the convention that the minimum over an empty set is $+\infty$.

We can summarize the optimal policy as follows. If only one queue is to be scheduled, then the optimal policy puts all b tasks on the shortest queue. If, on the other hand, both queues are to be scheduled then the optimal policy divides tasks among both queues in such a way that the *final configuration* of tasks makes the two queues as *even* as possible. In what follows, $\lfloor\;\rfloor$ and $\lceil\;\rceil$ denote the floor and ceiling functions, respectively ([21], p. 37).

Theorem 1. Suppose that in state $\mathbf{n}$, the optimal policy Ψ schedules $b \geq 1$ tasks. If either $b = 1$ or $1 \leq b \leq \Delta\mathbf{n}$, then **SQ** is necessarily satisfied. Otherwise,

1. If **SQ** is satisfied in state $\mathbf{n}$, then Ψ schedules all b tasks on queue 1.

2. If **BQ** is satisfied in state $\mathbf{n}$, then necessarily $2 \leq b$ and $\Delta\mathbf{n} < b$, and Ψ schedules $l_k(b;\mathbf{n})$ tasks on queue k, $k = 1,2$, with $l_k(b;\mathbf{n})$ given by

$$l_k(b;\mathbf{n}) \equiv \begin{cases} \Delta\mathbf{n} + \lfloor \frac{b-\Delta\mathbf{n}}{2} \rfloor, & k = 1, \\ \\ \lceil \frac{b-\Delta\mathbf{n}}{2} \rceil, & k = 2. \end{cases} , \tag{9}$$

Theorem 1 indicates how tasks are to be scheduled once the number of queues to schedule is determined. If **SQ** is satisfied then all tasks are scheduled on the shortest queue whereas if **BQ** is satisfied then tasks are assigned to each queue in a way that tends to equalize their final lengths or in other words to evenly balance the load across both processors.

If **SQ** is satisfied then it is clear that the minimum expected job response time is obtained when the shortest queue is assigned all b tasks and thus the expected response time is given by $ES^1_{n_1+b} = (n_1 + b)\overline{X}$. Combining Theorem 1 with (8), we thus see that **BQ** (resp. **SQ**) is satisfied in state $\mathbf{n}$ when scheduling b tasks provided

$$E \max_{k=1,2}\{S^k_{n_k+l_k(b;\mathbf{n})}\} \leq (\text{resp. } \geq) ES^1_{n_1+b} = (n_1 + b)\overline{X}. \tag{10}$$

In view of Claim 1 of Theorem 1 it seems more appropriate to read **SQ** as meaning *shortest* queue rather than *single* queue as previously defined and we adopt this nomenclature in the remainder of this section.

If **BQ** is satisfied in state $\mathbf{n}$, then the optimal policy allocates $l_k(b;\mathbf{n})$ tasks to queue k , $k = 1,2$, with $l_k(b;\mathbf{n})$ given by (9), and this corresponds to a load balancing property of the optimal policy: First $\Delta\mathbf{n}$ tasks are given to the shortest queue (queue 1) making it even with the largest queue (i.e. $n_1 + \Delta\mathbf{n} = n_2$). The remaining $b - \Delta\mathbf{n}$ tasks are then evenly distributed among the two queues, $\lfloor \frac{b-\Delta\mathbf{n}}{2} \rfloor$ being given to queue 1 and $\lceil \frac{b-\Delta\mathbf{n}}{2} \rceil$ to queue 2. As another way to look at this, we observe that the state $\mathbf{n}^* \equiv (n_1 + l_1(b;\mathbf{n}), n_2 + l_2(b;\mathbf{n}))$ resulting from the allocation (9) satisfies

$$\Delta\mathbf{n}^* = \lceil \frac{b-\Delta\mathbf{n}}{2} \rceil - \lfloor \frac{b-\Delta\mathbf{n}}{2} \rfloor = 0, 1. \tag{11}$$

3.2 Parallel versus Sequential Processing

Theorem 1 points to two policies, mentioned in the introduction, which are naturally associated with the conditions **BQ** and **SQ**, respectively. In terms of the underlying parallel processing model, these two policies correspond to sequential and parallel processing, respectively, and the tradeoff between these policies depends heavily on distributional properties of X and of the maximum operator. One might think that if the number of tasks were larger than the difference between the queue lengths, i.e., $b > \Delta\mathbf{n}$, then it would be beneficial to distribute some of the tasks to the larger queue thus gaining the potential benefits of parallel processing. This policy is optimal if $\sigma_X^2 = 0$ (i.e., if X is deterministic). However, as σ_X^2 increases, the maximum $\max_{k=1,2}\{S^k_{n_k+l_k(b;\mathbf{n})}\}$ also "increases" (i.e., becomes more variable) and therefore it might not be beneficial to incur potential delays associated with synchronizing with the second queue. Indeed, as stated in Theorem 3 below, for each value of b, there exists an integer $n^*(b)$ such that if the queue lengths satisfy $n_k > n^*(b), k = 1, 2$, then sequential processing is optimal. In practical systems, the r.v. B is bounded and for system loads resulting in large queues, Theorem 3 thus implies that the optimal scheduler would most frequently route complete jobs to the shortest queue and thus do very little parallel processing. This result runs counter to the intuition that parallel processing or load sharing is beneficial.

We first recall a monotonicity property of the **SQ** policy ([28], Lemma 4).

Lemma 2. Consider a state $\mathbf{n}$ with $n_1 \leq n_2$. If **SQ** is satisfied in state $\mathbf{n}$ when scheduling $b \geq 1$ tasks, then **SQ** is also satisfied in states $(n_1, n_2 + 1)$ and $(n_1 - 1, n_2)$ (with $n_1 \geq 1$) when scheduling these b tasks.

The second key structural property of the optimal scheduling policy Ψ is contained in the next proposition which follows essentially from an argument based on the Central Limit Theorem ([28], Theorem 5).

Theorem 3. Assume $\sigma_X^2 > 0$. For each integer $b \geq 1$ there exists a non-negative integer $n^*(b)$ such that **SQ** is optimal for scheduling b tasks when in states $\mathbf{n}$ satisfying $n_k \geq n^*(b)$, $k = 1, 2$.

Combining Lemma 2 and Theorem 3 provides a fairly complete picture of the form of the optimal policy Ψ. The value of $n^*(b)$ for the case where service times are exponentially distributed is determined by the solution to a simple algebraic equation ([28], Eqn. (50))

4 Some Preliminary Results

For an arbitrary number of processors, the structure of the optimal policy is far more complex than in the case $K = 2$. In particular, describing the load balancing properties of the optimal schedule requires a multi–dimensional version of Proposition A.1 from [28]. This extension, contained in Theorem 4 below, is developed later in this section, along with some additional auxilary results; they will be used in section 5 to determine the structure of the optimal policy. These results are best formalized with the help of notions of *majorization* and *stochastic ordering* which we now introduce.

We start with majorization; for additional information on this material we refer the reader to the monograph of Marshall and Olkin [29]: Given a positive integer $L \geq 2$, for any element $\mathbf{x}$ in $\mathbb{R}^L$, let

$$x_{(1)} \leq \cdots \leq x_{(L)} \tag{12}$$

denote the components $x_1, \ldots, x_L$ of $\mathbf{x}$ arranged in increasing order. For $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^L$, we say that $\mathbf{x}$ is majorized by $\mathbf{y}$ (in $\mathbb{R}^L$), and we write $\mathbf{x} \prec \mathbf{y}$, if

$$\sum_{i=1}^{l} x_{(i)} \geq \sum_{i=1}^{l} y_{(i)}, \quad l = 1, 2, \ldots, L-1, \tag{13}$$

and

$$\sum_{i=1}^{L} x_{(i)} = \sum_{i=1}^{L} y_{(i)}. \tag{14}$$

It is often convenient to interpret $\mathbf{x} \prec \mathbf{y}$ as saying that $\mathbf{x}$ is more balanced than $\mathbf{y}$. The binary relation on $\mathbb{R}^L$ defined by (13)–(14) is reflexive, transitive but not anti–symmetric, so that $\prec$ is a *preorder*, and not an order, on $\mathbb{R}^L$.

We also recall the definition of the increasing convex ordering on the collection of $\mathbb{R}$–valued r.v.'s [37, 39]: Let X and Y be two such r.v.'s. We say that the r.v. X is smaller than the r.v. Y in the increasing convex ordering, and we write $X \leq_{icx} Y$, if

$$E\phi(X) \leq E\phi(Y) \tag{15}$$

for every convex increasing mapping $\phi : \mathbb{R} \to \mathbb{R}$, provided the expectations in (15) exist.

4.1 A Stochastic Comparison Result

In this section we establish a key stochastic comparison result that already reveals some of the structural properties of the optimal schedule. In short, we show in Theorem 4, that if exactly L queues are scheduled, then the more balanced (as made precise by majorization) the state achieved by an allocation, the better the performance of the allocation.

Theorem 4. Let $\mathbf{n}$ and $\mathbf{m}$ be two elements of $\mathbb{N}^L$ for some arbitrary integer $L \geq 2$. If $\mathbf{n} \prec \mathbf{m}$ in $\mathbb{N}^L$, then we have

$$\max_{1 \leq l \leq L} \{S^l_{n_l}\} \leq_{icx} \max_{1 \leq l \leq L} \{S^l_{m_l}\} \tag{16}$$

and therefore

$$E \max_{1 \leq l \leq L} \{S^l_{n_l}\} \leq E \max_{1 \leq l \leq L} \{S^l_{m_l}\}. \tag{17}$$

To facilitate the discussion, we introduce for each integer $L \geq 2$ the notation

$$\mathrm{T}_L(\mathbf{n}) \equiv \max_{1 \leq l \leq L} \{S^l_{n_l}\}, \quad \mathbf{n} \in \mathbb{N}^L. \tag{18}$$

We can view the r.v. $\mathrm{T}_L(n_1, n_2, \ldots, n_L)$ as the job response time corresponding to an allocation of tasks to L queues which result in a state where queue l has n_l tasks, $l = 1, \ldots, L$.

We first consider the case $L = 2$ in the following proposition.

Theorem 5. Theorem 4 holds true for $L = 2$.

Proof. Theorem 5 will follow from the seemingly weaker Proposition A.1. proved in the appendix. Indeed, consider any two distinct elements $\mathbf{n}$ and $\mathbf{m}$ of $\mathbb{N}^2$ such that $\mathbf{n} \prec \mathbf{m}$; there is no loss of generality in assuming $n_1 \leq n_2$ and $m_1 < m_2$ with $m_1 < n_1$. The definition of majorization readily implies the existence of vectors $\mathbf{m}^{(s)}$, $s = 1, \ldots, r$, with $r = n_1 - m_1 + 1$, such that

$$\mathbf{n} = \mathbf{m}^{(r)} \prec \mathbf{m}^{(r-1)} \prec \ldots \prec \mathbf{m}^{(2)} \prec \mathbf{m}^{(1)} = \mathbf{m}. \tag{19}$$

For $s = 1, \ldots, r-1$, the vector $\mathbf{m}^{(s+1)}$ is generated from $\mathbf{m}^{(s)}$ by transferring one unit from the largest component of $\mathbf{m}^{(s)}$ to its smallest component, i.e., $\mathbf{m}^{(s+1)} = (m_1^{(s)}+1, m_2^{(s)}-1)$. It is then clear from Proposition A.1 that $\mathbf{T}_2(\mathbf{m}^{(s+1)}) \leq_{icx} \mathbf{T}_2(\mathbf{m}^{(s)})$, and the desired result follows from (19) in the form $\mathbf{T}_2(\mathbf{n}) \leq_{icx} \mathbf{T}_2(\mathbf{m})$ ■

In order to use Theorem 5 in establishing the general case, i.e., $L \geq 2$ arbitrary, we need to introduce the notion of a transfer operation on elements of $I\!R^L$: For given indices $i \neq j$ (with $i, j = 1, \ldots, L$), the *transfer* from i to j, or (i,j)–transfer, is the mapping $I\!R^L \to I\!R^L$ that maps a vector $\mathbf{x}$ in $I\!R^L$ into another vector $\mathbf{x}'$ of $I\!R^L$ given by $x_i' = x_i - 1$, $x_j' = x_j + 1$ and $x_k' = x_k$ otherwise, $k = 1, \ldots, L$. We call this vector $\mathbf{x}'$ the (i,j)–transfer of $\mathbf{x}$.

This notion of transfer has already been invoked in the proof of Theorem 5 in the form of $(2,1)$–transfers (on $I\!N^2$).

Lemma 6. Consider a pair of indices $i \neq j$ (with $i, j = 1, \ldots, L$), and let $\mathbf{n}$ be an element of $I\!N^L$ such that $n_i \geq 1$. Then the (i,j)–transfer $\mathbf{n}'$ of $\mathbf{n}$ is also an element of $I\!N^L$, and we have $\mathbf{n}' \prec \mathbf{n}$ if and only if $n_j < n_i$, in which case

$$\mathbf{T}_L(\mathbf{n}') \leq_{icx} \mathbf{T}_L(\mathbf{n}). \tag{20}$$

Proof. The first part follows readily from the definition of majorization. Now, let $\mathbf{n}$ be an element of $I\!N^L$ such that $0 \leq n_j < n_i$ for some pair of indices $i \neq j$ (with $i, j = 1, \ldots, L$). By the first part of the proof, the (i,j)–transfer $\mathbf{n}'$ of $\mathbf{n}$ is an element of $I\!N^L$ such that $\mathbf{n}' \prec \mathbf{n}$ (in $I\!N^L$). Applying the two–dimensional result of Proposition A.1 to the vector (n_i, n_j), we obtain

$$\max\{S^i_{n_i+1}, S^j_{n_j-1}\} \leq_{icx} \max\{S^i_{n_i}, S^j_{n_j}\}. \tag{21}$$

To proceed, we use the representations

$$\mathbf{T}_L(\mathbf{n}) = \max\left\{\max_{1 \leq l \leq L, l \neq i,j}\{S^l_{n_l}\}, \max\{S^i_{n_i}, S^j_{n_j}\}\right\} \tag{22}$$

and

$$\mathbf{T}_L(\mathbf{n}') = \max\left\{\max_{1 \leq l \leq L, l \neq i,j}\{S^l_{n_l}\}, \max\{S^i_{n_i+1}, S^j_{n_j-1}\}\right\} \tag{23}$$

with the observation that each of the r.v.'s $\max\{S^i_{n_i}, S^j_{n_j}\}$ and $\max\{S^i_{n_i+1}, S^j_{n_j-1}\}$ is independent of the r.v. $\max_{1\leq l\leq L, l\neq i,j}\{S^l_{n_l}\}$ under the enforced statistical assumptions. A direct application of Proposition 8.5.4 of [37] yields (20) with the help of (21), (22) and (23) since the maximum operator is increasing convex in its arguments. ■

We are now ready to conclude the proof of Theorem 4.

Proof of Theorem 4. Let $\mathbf{n}$ and $\mathbf{m}$ be two elements of $\mathbb{N}^L$ such that $\mathbf{n} \prec \mathbf{m}$. Invoking a theorem of Muirhead ([29], Lemma D.1, p. 135), [32], we can assert the existence of vectors $\mathbf{m}^{(s)}$, $s = 1, \ldots, r$, such that (19) holds and such that for each $s = 1, \ldots, r-1$, the vector $\mathbf{m}^{(s+1)}$ is generated from $\mathbf{m}^{(s)}$ by a transfer. These facts and Lemma 6 complete the argument.

■

4.2 A Majorization Result

Theorem 4 suggests that the allocation resulting in the most balanced state will be optimal. In this section we show the existence of at least one allocation which achieves the most balanced state. The argument is a constructive one, and relies on properties of the scheduling policy that successively adds a task to the queue with the current minimal length.

To facilitate the presentation, we find it convenient to introduce several auxiliary notions: For each $b \geq 1$, an allocation of b tasks on the K queues is any element $\mathbf{b}$ of $\mathbb{N}^K$ such that $\sum_{k=1}^K b_k = b$. The collection of all such allocations of b tasks on the K queues is denoted by $\mathcal{A}(b)$. Moreover, for any non-empty subset I of $\{1, \ldots, K\}$, we define an *I-allocation* of b to be any allocation $\mathbf{b}$ in $\mathcal{A}(b)$ such that

$$b_k \geq 1, \quad k \in I, \quad b_k = 0, \quad k \notin I. \tag{24}$$

We note that $\mathcal{A}_I(b)$ reduces to a singleton if $|I| = 1$, and is empty if $b < |I|$; moreover, for $K = 2$ we have $\mathcal{A}_{\{1,2\}}(b) = \mathcal{B}(b)$.

Next, for $k = 1, \ldots, K$, let $\mathbf{e}^k$ denote the element of $\mathbb{N}^K$ with components all zero except for the k^{th} one which is 1. We set

$$i(\mathbf{n}) \equiv \min\{k : 1 \leq k \leq K, n_k = \min\{n_1, \ldots, n_K\}\}, \quad \mathbf{n} \in \mathbb{N}^K \tag{25}$$

and define

$$\mu(\mathbf{n}) \equiv \mathbf{e}^{i(\mathbf{n})}, \quad \mathbf{n} \in I\!N^K. \tag{26}$$

The vector $\mu(\mathbf{n})$ can be interpreted as the element of $\mathcal{A}(1)$ which assigns a single task on the smallest queue with smallest index. Starting with $\mathbf{n}$ in $I\!N^K$, we now define recursively the states $\{\mathbf{n}^{(s)},\ s = 0, 1, \ldots\}$ by

$$\mathbf{n}^{(0)} = \mathbf{n}, \ \mathbf{n}^{(s+1)} = \mathbf{n}^{(s)} + \mu(\mathbf{n}^{(s)}), \ \ s = 0, 1, \ldots \tag{27}$$

The state $\mathbf{n}^{(s)}$ is the state that results from allocating s tasks on K queues, one task at a time on the smallest queue with smallest index. As we shall see, in some sense this procedure yields the optimal allocations. To do so, we first present two preliminary results on the single task allocations (26).

The first such result is contained in Lemma 7.

Lemma 7. For any element $\mathbf{n}$ of $I\!N^K$, we have

$$\mathbf{n} + \mu(\mathbf{n}) \prec \mathbf{n} + \mathbf{e}^j, \quad j = 1, 2, \ldots, K. \tag{28}$$

Proof. Without any loss of generality, we can assume that the components of $\mathbf{n}$ are ordered with $n_1 \le n_2 \le \ldots \le n_K$, in which case (28) is equivalent to $\mathbf{n} + \mathbf{e}^1 \prec \mathbf{n} + \mathbf{e}^j$ for all $j = 1, 2, \ldots, K$. These comparisons are easy consequences of the definition (13)–(14), with the proof left to the interested reader. ■

The second preliminary result is discussed in Lemma 8; its proof is given in Appendix B.

Lemma 8. Let $\mathbf{n}$ and $\mathbf{m}$ be elements of $I\!N^K$. Whenever $\mathbf{n} \prec \mathbf{m}$, we have the comparison

$$\mathbf{n} + \mu(\mathbf{n}) \prec \mathbf{m} + \mu(\mathbf{m}). \tag{29}$$

■

We conclude this section with a key result on the structure of the set of all possible states $\{\mathbf{n} + \mathbf{b},\ \mathbf{b} \in \mathcal{A}(b)\}$ that result from allocating b tasks on K processors when the system is in state $\mathbf{n}$.

Theorem 9. For every state $\mathbf{n}$ in $I\!N^K$, we have the comparison result

$$\mathbf{n}^{(b)} \prec \mathbf{n} + \mathbf{b}, \quad \mathbf{b} \in \mathcal{A}(b), \quad b = 1, 2, \ldots \tag{30}$$

Proof. The proof proceeds by induction: If $b = 1$, then (30) follows directly from Lemma 7, thus establishing the basis step. Now, assuming that (30) is true for some $b \geq 1$, we shall show that (30) also holds true for $b+1$. Let $\mathbf{b}$ be an arbitrary allocation in $\mathcal{A}(b)$. By the induction hypothesis, we have $\mathbf{n}^{(b)} \prec \mathbf{n} + \mathbf{b}$, so that

$$\mathbf{n}^{(b+1)} = \mathbf{n}^{(b)} + \mu(\mathbf{n}^{(b)}) \prec \mathbf{n} + \mathbf{b} + \mu(\mathbf{n} + \mathbf{b}) \tag{31}$$

upon invoking Lemma 8, while Lemma 7 implies

$$\mathbf{n} + \mathbf{b} + \mu(\mathbf{n} + \mathbf{b}) \prec \mathbf{n} + \mathbf{b} + \mathbf{e}^j, \quad j = 1, \ldots, K. \tag{32}$$

Combining (31)–(32), we see that

$$\mathbf{n}^{(b+1)} \prec \mathbf{n} + \mathbf{b} + \mathbf{e}^j, \quad j = 1, \ldots, K \tag{33}$$

and the induction step readily follows once we observe that $\{\mathbf{b} + \mathbf{e}^j,\ j = 1, 2, \ldots, K,\ \mathbf{b} \in \mathcal{A}(b)\}$ coincides with $\mathcal{A}(b+1)$. ■

Theorem 9 expresses the fact that the set $\{\mathbf{n} + \mathbf{b},\ \mathbf{b} \in \mathcal{A}(b)\}$ has a "root" or minimal element with respect to the preorder $\prec$. However, it is noteworthy that any two vectors in this set are not necessarily comparable under majorization as the following example indicates: For $K = 3$, consider the situation where $\mathbf{n} = (1, 2, 3)$ and $b = 16$. The allocations $\mathbf{b} = (2, 6, 8)$ and $\mathbf{b}' = (3, 4, 9)$ are both in $\mathcal{A}(16)$, yet the resulting states $\mathbf{n} + \mathbf{b} = (3, 8, 11)$ and $\mathbf{n} + \mathbf{b}' = (4, 6, 12)$ are not comparable under majorization since $4 > 3$ but $4 + 6 = 10 < 11 = 3 + 8$. Here the root is $(8, 7, 7)$ and we indeed have $(8, 7, 7) \prec (3, 8, 11)$ and $(8, 7, 7) \prec (4, 6, 12)$.

5 General K – Properties of the Optimal Policy

We are now ready to characterize the structure of the optimal policy for an arbitrary number of processors. This will done in several steps. First, on the basis of Theorems 4 and 9, we shall identify in section 5.1 a natural candidate policy for optimality. As this candidate policy may be suboptimal, we explore in section 5.2 its structure and find ways to improve its performance, thus exposing in the process the load balancing prooperties of optimal policies. In section 5.3, we summarize all the structural properties of the optimal policy.

5.1 A Near–Optimal Policy

We begin the discussion by restating Theorem 9 in the form

$$\mathbf{n}^{(b)} \prec \mathbf{n} + \pi(b;\mathbf{n}), \quad \mathbf{n} \in \mathbb{N}^K, \quad b = 1,2,\ldots, \quad \pi \in \mathcal{P}. \tag{34}$$

On the other hand, upon iterating (27), we obtain

$$\mathbf{n}^{(b)} = \mathbf{n} + \sum_{s=0}^{b-1} \mu(\mathbf{n}^{(s)}), \quad \mathbf{n} \in \mathbb{N}^K, \quad b = 1,2,\ldots \tag{35}$$

and this leads to the introduction of the scheduling policy γ in $\mathcal{P}$ defined by

$$\gamma(b;\mathbf{n}) \equiv \sum_{s=0}^{b-1} \mu(\mathbf{n}^{(s)}), \quad b = 1,2,\ldots, \quad \mathbf{n} \in \mathbb{N}^K. \tag{36}$$

The policy γ tends to equalize queue lengths since it sequentially allocates a task to the queue having the current minimal length.

Now, with Theorem 4 in mind, we might suspect from (34) that the scheduling policy γ exhibits some form of optimality. We explore this in Theorem 10 below. For each admissible policy π in $\mathcal{P}$, we define

$$L_\pi(b;\mathbf{n}) \equiv |I_\pi(b;\mathbf{n})|, \quad b = 1,2,\ldots, \quad \mathbf{n} \in \mathbb{N}^K \tag{37}$$

where

$$I_\pi(b;\mathbf{n}) \equiv \{k \in \{1,\ldots,K\} : \pi_k(b;\mathbf{n}) > 0\}, \quad b = 1,2,\ldots, \quad \mathbf{n} \in \mathbb{N}^K. \tag{38}$$

In words, $L_\pi(b;\mathbf{n})$ is the number of queues scheduled by policy π.

Theorem 10. The scheduling policy γ defined by (36) is optimal in the following sense: For any admissible policy π in $\mathcal{P}$ such that

$$L_\gamma(b;\mathbf{n}) \leq L_\pi(b;\mathbf{n}), \quad b = 1,2,\ldots, \quad \mathbf{n} \in \mathbb{N}^K, \tag{39}$$

we have

$$\overline{T}_\gamma(b;\mathbf{n}) \leq \overline{T}_\pi(b;\mathbf{n}), \quad b = 1,2,\ldots, \quad \mathbf{n} \in \mathbb{N}^K. \tag{40}$$

Proof. Fix the state $\mathbf{n}$ in $\mathbb{N}^K$ and the number $b \geq 1$ of tasks to be scheduled on the K queues. Obviously the result (39) will follow if for any subset I of $\{1, \ldots K\}$ with $|I| \geq L_\gamma(b;\mathbf{n})$, we can show that

$$\max_{k \in I_\gamma(b;\mathbf{n})} \{S^k_{n_k+\gamma_k(b;\mathbf{n})}\} \leq_{icx} \max_{k \in I} \{S^k_{n_k+b_k}\}, \quad \mathbf{b} \in \mathcal{A}_I(b). \tag{41}$$

In proving (41), there is no loss of generality in assuming that the components of the state $\mathbf{n}$ are arranged in increasing order, i.e., $n_1 \leq n_2 \leq \ldots \leq n_K$, as we do from now on in this proof. In that case, it follows from Lemma C.1 that (41) need only be established for subsets I of the form $\{1, \ldots, L\}$ with $L_\gamma(b;\mathbf{n}) \leq L \leq K$. We also note that, by construction, $I_\gamma(b;\mathbf{n}) = \{1, \ldots, L_\gamma(b;\mathbf{n})\}$.

Now, let I be any subset of the form $\{1, \ldots, L\}$ with $L_\gamma(b;\mathbf{n}) \leq L \leq K$. By Theorem 9, we have

$$\mathbf{n} + \gamma(b;\mathbf{n}) \prec \mathbf{n} + \mathbf{b}, \quad \mathbf{b} \in \mathcal{A}_I(b). \tag{42}$$

For every b in $\mathcal{A}_I(b)$, since $\gamma_l(b;\mathbf{n}) = b_l = 0$ for $l = L+1, \ldots, K$, we see that (42) is equivalent to

$$(n_1 + \gamma_1(b;\mathbf{n}), \ldots, n_L + \gamma_L(b;\mathbf{n})) \prec (n_1 + b_1, \ldots, n_L + b_L) \quad \text{in } \mathbb{N}^L, \tag{43}$$

and upon applying Theorem 4 (in conjunction with (43)), we obtain

$$\max_{k \in I} \{S^k_{n_k+\gamma_k(b;\mathbf{n})}\} \leq_{icx} \max_{k \in I} \{S^k_{n_k+b_k}\}, \quad \mathbf{b} \in \mathcal{A}_I(b). \tag{44}$$

On the other hand, the assumption $L_\gamma(b;\mathbf{n}) \leq L$ and the non-negativity of the involved r.v.'s readily imply

$$\max_{k=1,\ldots,L_\gamma(b;\mathbf{n})} \{S^k_{n_k+\gamma_k(b;\mathbf{n})}\} \leq \max_{k \in I} \{S^k_{n_k+\gamma_k(b;\mathbf{n})}\} \tag{45}$$

and the desired conclusion (41) follows upon combining (44)-(45). ■

Theorem 10 can be restated as follows: When in state $\mathbf{n}$, the policy γ allocates b tasks on *exactly* $L_\gamma(b;\mathbf{n})$ queues, and this allocation is best among all allocations that schedule these b tasks on *at least* as many queues. However, it should be emphasized that the policy γ does not necessarily determine the best allocation in *all* possible states $\mathbf{n}$. Indeed, it cannot be inferred from Theorem 10 that allocating the b tasks on fewer than $L_\gamma(b;\mathbf{n})$ queues is *not* optimal.

The (possible) suboptimality of the policy γ can be traced back to the fact that (41) derives essentially from the comparison (30), a purely algebraic property which makes no account for the variability of the service time distribution! However, Theorem 10 does point the way to a natural characterization of the optimal policy Ψ. Before developing this characterization, we present a useful result which is an immediate consequence of Lemma C.1.

Lemma 11. Suppose that in state $\mathbf{n}$, the optimal policy Ψ schedules $b \geq 1$ tasks on the queues with indices in I. Then the I–allocation $\Psi(b; \mathbf{n})$ can always be selected so that all b tasks are allocated on the $|I|$ shortest queues.

From now on we shall assume that the optimal schedule Ψ is of the form given in Lemma 8. We note that by construction the policy γ also enjoys this property.

5.2 Load Balancing Properties

We shall complement Theorem 10 by Theorem 13 below. This latter result expresses the load balancing property of the optimal policy, i.e., if the optimal policy Ψ schedules a subset of the queues, it does so in a way that tends to equalize their queue lengths. Before discussing Theorem 13, we extend the notation introduced in section 4.2, in order to more easily describe $\Psi(b; \mathbf{n})$ once it is known in state $\mathbf{n}$ on which queues to allocate the b tasks. Let I be a non–empty subset of $\{1, \ldots, K\}$. We set

$$i_I(\mathbf{n}) \equiv \min\{k \in I : n_k = \min\{n_i, \ i \in I\}\}, \quad \mathbf{n} \in I\!N^K \tag{46}$$

and define

$$\mu_I(\mathbf{n}) \equiv \mathbf{e}^{i_I(\mathbf{n})}, \quad \mathbf{n} \in I\!N^K. \tag{47}$$

We see that $\mu_I(\mathbf{n})$ is the element of $\mathcal{A}(1)$ which allocates a single task on the smallest queue with smallest index in I. Starting with $\mathbf{n}$ in $I\!N^K$, we now define recursively the states $\{\mathbf{n}^{(I,s)}, \ s = 0, 1, \ldots\}$ by

$$\mathbf{n}^{(I,0)} = \mathbf{n}, \ \mathbf{n}^{(I,s+1)} = \mathbf{n}^{(I,s)} + \mu_I(\mathbf{n}^{(I,s)}), \ \ s = 0, 1, \ldots \tag{48}$$

and the state $\mathbf{n}^{(I,s)}$ is thus the state that results from allocating s tasks on queues with index in I, one task at a time on the smallest queue with smallest index in I. In analogy with (36), we introduce the allocations

$$\gamma^I(b; \mathbf{n}) \equiv \sum_{s=0}^{b-1} \mu_I(\mathbf{n}^{(I,s)}), \quad b = 1, 2, \ldots, \quad \mathbf{n} \in I\!N^K. \tag{49}$$

Note that $i_I(\mathbf{n})$, $\mu_I(\mathbf{n})$ and $\gamma^I(b;\mathbf{n})$ reduce to $i(\mathbf{n})$, $\mu(\mathbf{n})$ and $\gamma(b;\mathbf{n})$, respectively, when $I = \{1,\ldots,K\}$. Alternatively, we can think of the definitions (46), (47) and (49) as being the definitions (25), (26) and (36) but on $\mathbb{N}^{|I|}$, rather than on $\mathbb{N}^K$.

Among other things, Theorem 12 presents conditions under which the allocations (49) are I–allocations. To that end, for each $\mathbf{n}$ in $\mathbb{N}^K$, we find it convenient to define

$$\Delta_k\mathbf{n} \equiv n_{(k+1)} - n_{(k)}, \quad k = 1,2,\ldots K-1 \tag{50}$$

and we set

$$\Gamma_k\mathbf{n} \equiv \sum_{l=1}^{k} l\Delta_l\mathbf{n}, \quad k = 1,2,\ldots,K-1 \tag{51}$$

with $\Gamma_0\mathbf{n} \equiv 0$. To lend an interpretation to (51) we rewite it as

$$\Gamma_k\mathbf{n} = \sum_{l=1}^{k} n_{(k+1)} - n_{(l)}, \quad k = 1,2,\ldots,K-1 \tag{52}$$

which shows that $\Gamma_k\mathbf{n}$ is the total number of tasks required to make each of the k shortest queues have queue lengths equal to $n_{(k+1)}$ tasks.

Theorem 12. Suppose that in state $\mathbf{n}$, with $n_1 \le n_2 \le \ldots \le n_K$, it is required to schedule $b \ge 1$ tasks on K queues. For any non-empty subset I of the form $\{1,\ldots,L\}$ for some $1 \le L \le K$, the following statements are true.

1. The allocation $\gamma^I(b;\mathbf{n})$ is an I–allocation if and only if

$$\Gamma_{L-1}\mathbf{n} + L \le b; \tag{53}$$

2. Under (53), we have

$$\mathbf{n} + \gamma^I(b;\mathbf{n}) \prec \mathbf{n} + \mathbf{b}, \quad \mathbf{b} \in \mathcal{A}_I(b) \tag{54}$$

and therefore

$$\max_{k\in I}\{S^k_{n_k+\gamma^I_k(b;\mathbf{n})}\} \le_{icx} \max_{k\in I}\{S^k_{n_k+b_k}\}, \quad \mathbf{b} \in \mathcal{A}_I(b) \tag{55}$$

3. If (53) does not hold, then there exists a non-empty subset J of I such that $\gamma^I(b;\mathbf{n})$ is a J-allocation of b, with $\gamma^I(b;\mathbf{n}) = \gamma^J(b;\mathbf{n})$. Moreover, the set J is of the form $\{1,\ldots,\hat{L}(b;\mathbf{n})\}$, where

$$\hat{L}(b;\mathbf{n}) \equiv \max\{l : 1 \leq l \leq K,\ \Gamma_{l-1}\mathbf{n} + l \leq b\}, \quad b = 1,2,\ldots \tag{56}$$

and this time we have

$$\max_{k\in J}\{S^k_{n_k+\gamma^J_k(b;\mathbf{n})}\} \leq_{icx} \max_{k\in I}\{S^k_{n_k+b_k}\}, \quad \mathbf{b} \in \mathcal{A}_I(b). \tag{57}$$

Proof. Since we assume $n_1 \leq n_2 \leq \ldots \leq n_K$, we get $\Delta_k\mathbf{n} = n_{k+1} - n_k$ for $k = 1,2,\ldots K-1$.

(Claim 1) That (53) is a necessary and sufficient condition for the allocation $\gamma^I(b;\mathbf{n})$ to be an I-allocation follows from (46)-(49) by direct inspection.

(Claim 2) Under (53), $\gamma^I(b;\mathbf{n})$ is an I-allocation by Claim 1, so that

$$\gamma^I_l(b;\mathbf{n}) = b_l = 0, \quad l = L+1,\ldots,K, \quad \mathbf{b} \in \mathcal{A}_I(b). \tag{58}$$

It is now clear from the definitions (46)-(49) (and remarks following (49)) that Theorem 9 (when interpreted in $I\!N^L$) implies the comparison

$$(n_1 + \gamma^I_1(b;\mathbf{n}),\ldots,n_L + \gamma^I_L(b;\mathbf{n})) \prec (n_1 + b_1,\ldots,n_L + b_L), \quad \mathbf{b} \in \mathcal{A}_I(b) \quad (\text{ in } I\!N^L). \tag{59}$$

The comparison (54) follows by combining (58) and (59) by means of an elementary concatenation property ([29], Prop. A.7, p. 121). The stochastic comparison (55) is now obtained by applying Theorem 4 with (59).

(Claim 3) If (53) fails, then by Claim 1 we already know that $\gamma^I(b;\mathbf{n})$ is not an I-allocation of the b tasks, as it assigns them on fewer than L queues. In fact, it is straightforward from (46)-(49) that $\gamma^I(b;\mathbf{n})$ assigns the b tasks on the $\hat{L}(b;\mathbf{n})$ smallest queues; is is clear that $\hat{L}(b;\mathbf{n}) < L$ under the enforced assumptions. With $J \equiv \{1,\ldots,\hat{L}(b;\mathbf{n})\}$, we obviously get that $\gamma^I(b;\mathbf{n})$ is now a J-allocation, and the identification $\gamma^I(b;\mathbf{n}) = \gamma^J(b;\mathbf{n})$ readily takes place. To prove (57), it suffice to notice that (54) is still valid, whence (55) also holds. We conclude that (57) follows from these remarks upon observing that

$$\max_{k\in J}\{S^k_{n_k+\gamma^J_k(b;\mathbf{n})}\} \leq \max_{k\in I}\{S^k_{n_k+\gamma^I_k(b;\mathbf{n})}\}, \tag{60}$$

an obvious fact given that the involved r.v.'s are non-negative and $|J| < |I|$. ■

5.3 Properties of the Optimal Policy

First we present the load balancing properties of the optimal policy Ψ.

Theorem 13. Suppose that in state $\mathbf{n}$, with $n_1 \leq n_2 \leq \ldots \leq n_K$, it is required to schedule $b \geq 1$ tasks on K queues. The optimal allocation $\Psi(b;\mathbf{n})$ can always be selected to be an I^*-allocation for some non-empty subset I^* of the form $\{1,\ldots,L^*\}$ with $1 \leq L^* \leq K$, such that the condition (53) necessarily holds. In that case, the allocation $\Psi(b;\mathbf{n})$ can be chosen to be $\gamma^{I^*}(b;\mathbf{n})$, so that

$$n^* \equiv \mathbf{n} + \Psi(b;\mathbf{n}) \prec \mathbf{n} + \mathbf{b}, \quad \mathbf{b} \in \mathcal{A}_{I^*}(b) \tag{61}$$

with

$$n_1^* \geq \ldots \geq n_{L^*}^*. \tag{62}$$

Furthermore, we have the bound $1 \leq L^* \leq \hat{L}(b;\mathbf{n})$.

Proof. Fix the state $\mathbf{n}$ in $\mathbb{N}^K$ and the number $b \geq 1$ of tasks to be scheduled on the K queues. Since $n_1 \leq \ldots \leq n_K$, by Lemma 11 (and the remark following it), we can assume $\Psi(b;\mathbf{n})$ to be an I-allocation with I a subset of the form $\{1,\ldots,L\}$ for some $1 \leq L \leq K$. If $\Gamma_{L-1}\mathbf{n} + L \leq b$, then by Claim 1 of Theorem 12, the allocation $\gamma^I(b;\mathbf{n})$ is indeed an I-allocation and by Claim 2 of Theorem 12, it is at least as good as the optimal allocation $\Psi(b;\mathbf{n})$. Therefore, we can take $\Psi(b;\mathbf{n}) = \gamma^I(b;\mathbf{n})$ so that $I^* = I$ and $L^* = L$. We observe that (61) follows from (54), and that $1 \leq L \leq \hat{L}(b;\mathbf{n})$ obviously holds.

On the other hand, if $\Gamma_{L-1}\mathbf{n} + L > b$, then $\gamma^I(b;\mathbf{n})$ is not an I-allocation by Claim 1 of Theorem 12. However, by Claim 3 of Theorem 12, with $J = \{1,\ldots,\hat{L}(b;\mathbf{n})\}$, the identification $\gamma^I(b;\mathbf{n}) = \gamma^J(b;\mathbf{n})$ holds and $\gamma^J(b;\mathbf{n})$ is now a J-allocation. We conclude from (57) that $\gamma^J(b;\mathbf{n})$ is at least as good as the allocation $\Psi(b;\mathbf{n})$. Therefore, we can take $\Psi(b;\mathbf{n}) = \gamma^J(b;\mathbf{n})$ so that $I^* = J$ and $L^* = \hat{L}(b;\mathbf{n})$. Here again (61) follows from (54), while $L^* = \hat{L}(b;\mathbf{n})$ by construction.

The inequalities (62) are now immediate by construction. ■

From the discussion given above, we see that the determination of the optimal policy proceeds in two steps. First the policy must determine the set I of queues on which the tasks will be allocated. Once this optimal set of queues has been identified, the optimal I-allocation is determined by (49). The procedure for selecting the optimal set of queues is easily obtained by combining Theorems 10 and 12. Indeed, for every subset I of $\{1, \ldots, K\}$, we define

$$T_I(b; \mathbf{n}) \equiv E \max_{k \in I} \{S^k_{n_k + \gamma^I_k(b;\mathbf{n})}\}, \quad b = 1, 2, \ldots, \quad \mathbf{n} \in I\!N^K. \tag{63}$$

Now suppose that in state $\mathbf{n}$, it is required to schedule $b \geq 1$ tasks on the K queues. We can always relabel the queues so that $n_1 \leq \ldots \leq n_K$, and therefore by Lemma 11 we need only consider sets of the form $\{1, \ldots, L\}$ with $1 \leq L \leq K$. By Theorem 13, we can further reduce the search to sets of the form $\{1, \ldots, L\}$ with $1 \leq L \leq \hat{L}(b; \mathbf{n})$. The optimal set of queues is the subset $\{1, \ldots, L^*(b; \mathbf{n})\}$ where

$$L^*(b; \mathbf{n}) \equiv \arg\min\{1 \leq L \leq \hat{L}(b; \mathbf{n}) : T_{\{1,\ldots,L\}}(b; \mathbf{n})\}; \tag{64}$$

ties are resolved by taking the smallest integer. Once the value of $L^*(b; \mathbf{n})$ has been computed, Theorem 12 then states that the optimal policy will successively schedule tasks to the shortest queue with index in $\{1, \ldots, L^*(b; \mathbf{n})\}$.

This discussion is summarized in the following corollary.

Corollary 14. Consider a state $\mathbf{n}$, with $n_1 \leq n_2 < \ldots \leq n_K$, and suppose that $b \geq 1$. Then

1. If $1 \leq b < \Gamma_1 \mathbf{n} + 2$, then the optimal policy Ψ schedules all b tasks on queue 1;
2. If for some $L = 1, \ldots, K-1$, $\Gamma_{L-1}\mathbf{n} + L \leq b < \Gamma_L \mathbf{n} + L + 1$, then the optimal policy Ψ schedule all b tasks on no more than the L first queues.

Proof. These statements are direct consequences of Theorem 13, and the discussion that follows it once we observe that for $L = 1, \ldots, K-1$, $\hat{L}(b; \mathbf{n}) = L$ if and only if $\Gamma_{L-1}\mathbf{n} + L \leq b < \Gamma_L \mathbf{n} + L + 1$. ■

In analogy with Theorem 5 of [28], we have the following result.

Theorem 15. For each integer $b \geq 1$ there exists a non-negative integer $n^*(b, K)$ such that if $n_k \geq n^*(b, K), k = 1, 2, \ldots, K$, then the optimal policy schedules all b tasks on the queue of shortest length, i.e. queue $\imath(\mathbf{n})$.

Proof. The proof of this theorem is a generalization of that found in the $K = 2$ case of [28] and is omitted here.

■

6 Comparison Results and Discussion

In this section we present results from a set of simulation experiments which we performed in order to determine response time properties of the optimal scheduler. Throughout, we make the following assumptions: Arrivals to the system come from a Poisson point source with intensity λ. Task service times, or stages of service, are exponentially distributed with parameter μ; this assumption allows us, as mentioned in section 2, to avoid difficulties arising from residual service times. Moreover, the number of tasks composing each job is determined randomly, and this independently of other random events occurring in the system, past, present and future. Let B be the r.v. denoting the generic number of tasks composing a job, with $\beta_b \equiv P[B = b]$, $b = 1, 2, \ldots$; the expected number of tasks is then given by $\overline{B} = \sum_{b=1}^{\infty} b\beta_b$. Clearly, for this model, the utilization of the system is given by

$$\rho \equiv \frac{\lambda \overline{B}}{K\mu}. \tag{65}$$

All simulation and computational results displayed below are carried out in the generic case where $\mu = 1$, i.e., task service times are exponentially distributed with unit mean. We also assume throughout that $B \equiv b$ for some fixed integer b, in which case $\rho = \lambda b / K\mu$; we refer the reader to [28] for comparison results concerning the case of random job size B when $K = 2$. For an arbitrary number of processors, we have omitted discussing the more general case of random job size for the corresponding simulations become computationally prohibitive, and it is our belief that the insights derived in the case $K = 2$ can be extrapolated to the more general situation.

Our main interest in these simulation experiments consists in determining the performance improvement that the optimal policy yields over simpler policies and to determine the fraction of time the optimal policy uses multiple processors. The simple policies that we consider are the **AQ** policy, which corresponds to the usual practice of distributing the load evenly among all the processors, and the **SQ** policy. When considering the **AQ** policy, we always assume that $b = JK$ for some integer $J \geq 1$, and J tasks are scheduled on each one of the K queues. The information required by each of the policies differs in that the optimal policy requires knowledge of all the queue lengths whereas the **AQ** policy uses no knowledge at all and the **SQ** policy requires only knowing the queue(s) of minimal queue length. It is not too surprising that our results show each of the last two policies to have regions over which they perform well. What is somewhat surprising, however, is that the **AQ** policy can perform very poorly in high utilizations and this argues against the intuition that parallel processing is beneficial. In fact, as we will shortly see, a job is scheduled on only a small number of processors by the optimal schedule in medium to heavy loads.

In what follows, the performance of any one of these three scheduling policies is expressed through its expected response time in steady-state. A moment of reflection should convince the reader that under all three policies, statistical equilibrium will be realized under the same stability condition, namely $\rho < 1$, a condition always enforced thereafter. Determining the expected response time of the optimal schedule policy appears to be mathematically intractable. Even for the simple policies **AQ** and **SQ**, the expected response times can only be obtained by simulation or approximation; such approximations have been developed for the **AQ** policy (with $b = K$) in [35] and for the **SQ** policy in [36]. A special case of the **AQ** policy for $K = b = 2$ does, however, have an exact solution for the expected response time [16, 35] which is given by

$$T_2 = \frac{12\mu - \lambda}{8} \frac{1}{\mu - \lambda}. \tag{66}$$

All simulated values have 99% confidence intervals that are within 2% of the simulated value and we simulated for utilizations in the range (.01, .95) in increments of .01. To avoid a cluttered presentation, we do not show points or confidence intervals on the plots of this section, but join simulated values with straight lines. The smoothness of the curves presented in this section attests to the accuracy of the simulation points.

To determine the optimal policy, we need a method to calculate the expected value of the maximum of a set of independent sums of i.i.d. services times. In particular, we need to evaluate the quantities

$$f_k(\mathbf{n}) \equiv E \max_{1 \leq i \leq k} \{S^i_{n_i}\}, \quad \mathbf{n} \in I\!N^K, \quad k = 1, 2, \ldots, K, \tag{67}$$

where under the assumptions made, $S^i_{n_i}$ is an Erlang r.v. with mean n_i, $i = 1, \ldots, K$. We observe that $f_k(\mathbf{n})$ represents the expected response when scheduling (the first) k queues results in state $\mathbf{n}$. Although an integral expression for the quantity (67) can be derived [12], it is not computationally accurate and thus cannot be used for determining the optimal policy. A more accurate implementation is obtained by solving a set of recurrence relationships. To define these recurrences, we let

$$z_k(\mathbf{n}) \equiv \sum_{i=1}^{k} \delta(n_i), \quad \mathbf{n} \in I\!N^K, \quad k = 1, 2, \ldots, K, \tag{68}$$

where $\delta(x) = 1$ if $x > 0$ and $\delta(x) = 0$ if $x = 0$. The value of $z_k(\mathbf{n})$ is thus the number of nonzero queue lengths among the k first queues when the system is in state $\mathbf{n}$. For each $k = 1, 2, \ldots, K$, upon using the memoryless property of exponential distributions, we can write the following recurrence

$$f_k(\mathbf{n}) = \begin{cases} 0, & \mathbf{n} = \mathbf{0}, \\ \frac{1}{z_k(\mathbf{n})} \left(1 + \sum_{i=1}^{k} \delta(n_i) f_k(\mathbf{n} - \mathbf{e}^i)\right), & \textit{otherwise}. \end{cases} \tag{69}$$

To explain (69) we first note that if there are $z_k(\mathbf{n})$ servers busy then the first departure event occurs as the minimal of $z_k(\mathbf{n})$ exponentials and thus has an expectation of $1/z_k(\mathbf{n})$ time units This explains the first term. Additionally, since each sever is equally likely to be the first to finish, the probability of going from state $\mathbf{n}$ to state $\mathbf{n} - \mathbf{e}^i$ is given by $\delta(n_i)/z_k(\mathbf{n})$ leading to an additional delay of $f_k(\mathbf{n} - \mathbf{e}^i)$, thus explaining the next term in the recurrence. The recurrences (69) can be easily programmed although the memory requirements are quite large for large values of K and $\mathbf{n}$. In Table 1 we list the experiments that we ran.

Table 1.

Experiment	K and B	Distribution of Task Service
I	$K = 2, \ b = 2$	Exponential
II	$K = 4, \ b = 8$	Exponential
III	$K = 4, \ b = 4$	Erlang (5)
IV	$K = 2, 4, \ b = K$	Exponential

Experiment I considers $K = 2$ and a fixed number of arriving tasks to be $b = 2$. Other experiments for the case $K = 2$ that include cases where B is random can be found in [28].

We did not perform any experiments here for B random and higher values of K because of the high computational and memory requirements needed to perform experiments. It is also our belief that results for these cases will not qualitatively differ from those obtained from the two processor case. As mentioned above, the exact value for the expected response time of the **AQ** policy is available in that case, and is given by (66). Therefore, we need only to simulate the optimal policy and the **SQ** policy. For this case, it is interesting to notice that under the optimal policy, both queues are scheduled only when the queue lengths satisfy $n_1 = n_2 = n$ for $n = 0, 1, 2$. In all other states only the shortest queue is scheduled. In figure 2 we plot the expected response times as a function of ρ and in figure 3 we plot ratios of expected response times for both the **AQ** and **SQ** policies to that of the optimal policy. It is interesting to observe the regions where each one of these policies does well in comparison to the optimal policy. For low utilizations scheduling both queues is close to optimal whereas scheduling only one queue is close to optimal for high utilizations. The best way to explain these relationships is to plot the expected number of queues scheduled by the optimal policy (as done in figure 4). This shows how the optimal policy changes from a regime of using parallel processing for low loads to a regime of using sequential processing for high loads. Since the policies **AQ** and **SQ** mimic the operation of the optimal policy for low and high loads, respectively, it is not surprising that their expected response times are close for these regions. Based on ratios of expected response time only, it would be tempting to select the **SQ** policy over the **AQ** policy if one were restricted to these two policies. This is especially true if one considers that for low utilizations, the ratios are for low expected response times whereas for high utilizations, where the policy **SQ** does very well, these ratios are for large values. Typically in most systems, it is more important to have good performance for high utilizations since users of the system experience changes in absolute not relative response times and these absolute changes are low for small response times.

Experiment **II** collaborates our conclusions from the first experiment. Here we set $K = 4$ and $b = 8$ and figures 5–7 are analogous to those obtained previously. Again these figures suggest that, of the two non–optimal policies, the policy **SQ** performs relatively better than the policy **AQ** since it is closer to the optimal policy at high utilizations. Figure 7 suggests that the expected number of scheduled queues for the optimal policy decreases almost linearly until high utilizations where it rapidly decreases.

In Experiment **III** we compare the optimal policies obtained for exponential and Erlang distributions of task service times. It is clear that since Erlang random variables are simply sums of exponential random variables, that the optimal policy for this case can be obtained by scaling the optimal policy obtained for the exponential case. In figure 8 we plot the expected response times obtained for task distributions that are given by the sum of 5 exponentials each with mean 0.2, i.e., an Erlang r.v. with 5 stages and unit mean, for a system with $K = 4$ and $b = 4$. As expected, the system with Erlang task times has a lower expected response time than that for exponentials. Generally, we expect that response time increases with variance and the number of processors scheduled by the optimal policy decreases with variance. This latter effect can be seen in figure 9 which plots the expected number of processors scheduled by the optimal policy for both task distributions. Figures 10 and 11 compare the response times obtained for the **SQ** and **AQ** policies in the Erlang case. Again, if it is required to select a single policy for use at all utilizations, the **SQ** policy would be preferable to the **AQ** policy.

Experiment **IV**, as shown in Figure 12, compares the optimal policy to that of a single server system with a capacity equal to the number of servers. Specifically, when the distributed system has K servers, we consider a single server system which delivers K units of service per unit time, where jobs are executed completely on this single server. We assume that $b = K$, and this implies that the service time duration in the coalesced server is the sum of b i.i.d. exponential r.v.'s, and is thus an Erlang r.v. with b stages. With the help of the Pollaczek-Khinchin formula [20], the resulting response time can be written as

$$T_{M/E_b/1} = \frac{1}{K}\left(b - \frac{(b-1)}{2}\rho\right)\frac{1}{\mu(1-\rho)} \tag{70}$$

where $\rho = \lambda/K\mu$. The figure plots the ratio of this value to the expected response time for the optimal policy. Clearly the single coalesced server has better response time but it is interesting to see that the relative response time between this and the optimal policy improves with increasing K. This arises, we believe, from the fact that the optimal scheduler has more degrees of freedom in making scheduling decisions when K is larger and thus it has a relatively smaller response time with respect to the coalesced single server.

In figure 13 we plot values of $n^*(b, K)$ for $K = 2$ as a function of b. For queue lengths greater than $n^*(b, K)$ the optimal policy only schedules the shortest queue. The value of $n^*(b, K)$ is given by

$$n^*(b, K) = \min\{m \mid m + b \leq \overline{T}_{\Psi}(b; \mathbf{m})\}, \quad b = 1, 2, \ldots \tag{71}$$

where for each non–negative integer m, we denote by $\mathbf{m}$ the balanced queue length vector $(m, \ldots, m)$. The points in figure 13 are the calculated values of $n^*(b, K)$ and the curves are the quadratic polynomials that best fit the points in a least squares sense. This polynomial is given by $n^*(b, 2) \approx -.5 - .414b + .789b^2$.

7 Conclusions and Future Research

In this paper we derived the individually optimal scheduling policy for executing jobs consisting of a set of independent tasks on a set of homogeneous processors. We provided simulation results to compare different scheduling policies when task service times were exponentially distributed with unit mean. Under these assumptions, our results imply that (i) parallel processing is beneficial for low system utilizations, and that (ii) for moderate to high utilizations, processing all the tasks of a job on a single processor produces smaller expected response times. In particular, the policy that executes all the tasks of a job sequentially on the processor with minimal queue length has a better overall response time than the policy which spreads work evenly over all the processors in the system. Including overheads due to executing jobs in parallel in our model would only increase the benefits of sequential processing. Our model, which is biased towards parallel processing, therefore strongly indicates that for the architectural assumptions of distributed queues one should only run jobs in parallel in lighly loaded systems. This result runs counter to the intuition that parallel processing is always beneficial.

These conclusions depend upon the distribution of task service times and we believe that the exponential distribution is more variable than what is found in some applications. As a general rule, we expect that parallel processing to become more beneficial as the distribution becomes less variable. For example, if service times were deterministic then it would be beneficial to schedule the tasks of a job on all processors. We believe our results, however, point out the fallacy of assuming that parallel processing is always beneficial and believe that the trend of the optimal policy to go from parallel to sequential processing as the load of the system increases, will be found in applications that have less variable distributions as well.

Deriving the socially optimal policy is an interesting, open research problem. Although we do not expect the response time for this policy to be markedly better than that of the individually optimal policy derived here, it would be interesting to quantify this improvements. Finally, determining a way to account for residual service times in the derivation of the individually optimal policy is also of interest. Here difficult mathematical problems arise from the fact that residual times on the different processors may not be independent r.v.'s due to the nature of the scheduling policy.

Acknowledgements. We gratefully acknowledge Laurence Durivault's help in generating the simulation results found in the paper. We also thank C.S. Chang for useful discussions and for pointing out how his recent work in stochastic majorization could be used to prove Theorem 4 for general task service times.

This work was initiated while the first author was a summer visitor at the IBM Thomas J. Watson Research Center, Yorktown Heights. The gracious hospitality of the Systems Analysis Department of S. S. Lavenberg is gratefully acknowledged. Support provided by the Systems Research Center was made possible through the National Science Foundation's Engineering Research Centers Program under grant NSFD CDR 88-03012.

8 Appendices

The results in the appendices that follow contain technical results that are not essential for understanding the main conclusions of the paper.

8.1 Appendix A

The proof of Theorem 5 passes by the following intermediary result.

Proposition A.1. For any state $\mathbf{n}$ in $I\!N^2$ with $0 \leq n_1 < n_2$, we have

$$\max\{S^1_{n_1+1}, S^2_{n_2-1}\} <_{icx} \max\{S^1_{n_1}, S^2_{n_2}\}. \tag{72}$$

so that

$$E\max\{S^1_{n_1+1}, S^2_{n_2-1}\} \leq E\max\{S^1_{n_1}, S^2_{n_2}\}. \tag{73}$$

Although Proposition A.1 and its consequences given in section 4 do hold for general task distributions, we have elected to provide a proof of Proposition A.1 only in the case where task distributions are exponential, say with unit parameter, in which case

$$F_n(t) \equiv P[S_n^k \leq t] = 1 - \sum_{i=0}^{n-1} \frac{t^i e^{-t}}{i!}, \ \ t \geq 0, \ k = 1, 2, \ldots, n = 1, 2, \ldots \tag{74}$$

As previously mentioned, in this exponential case our results are exact since the distribution of residual times seen by an arriving job are identical to the distribution of service times. The general case can be established using techniques of stochastic majorization recently developed by Chang [7]. Presenting a proof along these lines would require substantial preliminary work which lies outside the scope of this paper. Our proof of the special case uses standard techniques and keeps the paper self contained. We also note that the case where task service times have an Erlang distribution readily follows from the validity of (73) in the exponential case.

Proof. With Theorem 1.3.1 of ([39], p. 9) in mind, we set

$$T_a(\mathbf{n}) \equiv E[\max(S_{n_1}^1, S_{n_2}^2) - a]^+, \ \ a \geq 0, \ \ \mathbf{n} \in I\!N^2 \tag{75}$$

so that (72) follows if we can show that

$$T_a(n_1 + 1, n_2 - 1) \leq T_a(n_1, n_2), \ \ a \geq 0 \tag{76}$$

whenever $0 \leq n_1 < n_2$.

We fix $a \geq 0$. With the help of (74), for each $\mathbf{n}$ in $I\!N^2$, we easily see that

$$\begin{aligned} T_a(\mathbf{n}) &= \int_a^\infty (1 - F_{n_1}(t) F_{n_2}(t)) dt \\ &= \int_a^\infty \left\{ \sum_{i=0}^{n_1-1} \frac{t^i}{i!} e^{-t} + \sum_{j=0}^{n_2-1} \frac{t^j}{j!} e^{-t} + \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} \frac{t^{i+j}}{i!j!} e^{-2t} \right\} dt \\ &= e^{-a} \left\{ \sum_{i=0}^{n_1-1} \sum_{k=0}^{i} \frac{a^k}{k!} + \sum_{j=0}^{n_2-1} \sum_{l=0}^{j} \frac{a^l}{l!} - \sum_{i=0}^{n_1-1} \sum_{j=0}^{n_2-1} e^{-a} \binom{i+j}{i} 2^{-(i+j+1)} \sum_{r=0}^{i+j} \frac{(2a)^r}{r!} \right\} \end{aligned} \tag{77}$$

where in the last step, we have used the well-known identities

$$\int_a^\infty \frac{t^i}{i!} e^{-t} dt = e^{-a} \sum_{k=0}^{i} \frac{a^k}{k!}, \ \ a \geq 0, \ i = 0, 1, \ldots. \tag{78}$$

for Erlang distributions [20]. We note that the expression just obtained for $T_a(\mathbf{n})$ also holds when components of $\mathbf{n}$ do vanish provided we adhere to the convention that summing over an empty set of terms incurs the value zero. This convention is implicitly enforced throughout in order to avoid discussing separately various limiting cases.

To proceed, we consider only the vectors $\mathbf{n}$ in $I\!N^2$ for which $0 \leq n_1 < n_2$. Writing the components of $\mathbf{n}$ as $n_1 = m$ and $n_2 = m + p$ with $m = 0, 1, \ldots$ and $p = 1, 2, \ldots$, we define

$$\Delta_a(m,p) \equiv e^a(T_a(m, m+p) - T_a(m+1, m+p-1)), \quad m = 0, 1, \ldots, \quad p = 1, 2, \ldots, \tag{79}$$

so that the desired inequalities (76) are now equivalent to

$$\Delta_a(m,p) \geq 0, \quad m = 0, 1, \ldots, \quad p = 1, 2, \ldots. \tag{80}$$

Using the expression (77) derived earlier for $T_a(\mathbf{n})$, we are led after some simplifications to the expression

$$\begin{aligned} \Delta_a(m,p) = \sum_{l=m+1}^{m+p-1} \frac{a^l}{l!} &+ e^{-a} \left\{ \sum_{i=0}^{m+p-2} \binom{m+i}{i} 2^{-(m+i+1)} \sum_{r=0}^{m+i} \frac{(2a)^r}{r!} \right\} \\ &- e^{-a} \left\{ \sum_{i=0}^{m-1} \binom{m+p+i-1}{i} 2^{-(m+p+i)} \sum_{r=0}^{m+p+i-1} \frac{(2a)^r}{r!} \right\} \end{aligned} \tag{81}$$

for all $m = 0, 1, \ldots$ and $p = 1, 2, \ldots$.

Fix $m = 0, 1, \ldots$. By direct inspection of (81) (with $p = 1$) we see that $\Delta_a(m, 1) = 0$ (as expected from the probabilistic interpretation of the involved quantities). Therefore, (80) will be established if we can show that $p \rightarrow \Delta_a(m,p)$ is non–decreasing, that is

$$\Delta_a(m, p+1) - \Delta_a(m,p) \geq 0, \quad p = 1, 2, \ldots. \tag{82}$$

To do this, we observe from (81) by straightforward calculations that

$$\begin{aligned} &2^{m+p} e^a \left(\Delta_a(m, p+1) - \Delta_a(m,p) \right) \\ = &\frac{(2a)^{m+p}}{(m+p)!} \left\{ e^a - \frac{1}{2} \sum_{i=0}^{m-1} \frac{a^i}{i!} \right\} + \frac{1}{2^m} \binom{2m+p-1}{m} \sum_{k=0}^{2m+p-1} \frac{(2a)^k}{k!} \\ + &\sum_{i=0}^{m-1} \frac{1}{2^{i+1}} \binom{m+p+i}{i} \left(\frac{2(m+p)}{m+p+i} - 1 \right) \sum_{j=0}^{m+p+i-1} \frac{(2a)^j}{j!}, \quad p = 1, 2, \ldots \end{aligned} \tag{83}$$

The second term in the right handside of (83) is obviously positive, and so is the third term since

$$\frac{2(m+p)}{m+p+i} - 1 = \frac{m+p-i}{m+p+i} > 0, \quad i = 0, \ldots, m-1. \tag{84}$$

Therefore, we can conclude to (82) if the first term in the right handside of (83) is non–negative, a requirement equivalent to

$$\sum_{i=0}^{m-1} \frac{a^i}{i!} \leq 2e^a. \tag{85}$$

This last condition is trivially true as can be seen from the Taylor series expansion of e^a since $a \geq 0$, so that (82) indeed follows and (80) is thus established. ■

8.2 Appendix B

We provide here the proof of Lemma 8. There is no loss of generality in assuming $n_1 \leq n_2 \leq \ldots \leq n_K$ and $m_1 \leq m_2 \leq \ldots \leq m_K$, in which case (29) is equivalent to

$$\mathbf{n} + \mathbf{e}^1 \prec \mathbf{m} + \mathbf{e}^1. \tag{86}$$

¿From the definition (13)–(14) of majorization, we recall that the condition $\mathbf{n} \prec \mathbf{m}$ is equivalent to

$$\sum_{i=1}^{k} (n_i - m_i) \geq 0, \quad l = 1, 2, \ldots, K-1 \tag{87}$$

with

$$\sum_{i=1}^{K} n_i = \sum_{i=1}^{K} m_i \equiv C. \tag{88}$$

We now define the integers α and β by $\alpha \equiv \max\{k : 1 \leq k \leq K, \ n_1 = \ldots = n_k\}$ and $\beta \equiv \max\{k : 1 \leq k \leq K, \ m_1 = \ldots = m_k\}$. We note that $n_\alpha < n_{\alpha+1}$ whenever $\alpha < K$, with a similar comment for β. If $\mathbf{a}$ and $\mathbf{b}$ denote the elements of $I\!N^K$ obtained by arranging the components of $\mathbf{n} + \mathbf{e}^1$ and $\mathbf{m} + \mathbf{e}^1$ in increasing order, respectively, we readily see that

$$\mathbf{a} = (\underbrace{n_1, n_1, \ldots, n_1}_{\alpha-1}, n_\alpha + 1, n_{\alpha+1}, \ldots, n_K) \tag{89}$$

and

$$\mathbf{b} = (\underbrace{m_1, m_1, \ldots, m_1}_{\beta-1}, m_\beta + 1, m_{\beta+1}, \ldots, m_K). \tag{90}$$

¿From the definition (13)–(14) of majorization, we see that in order to establish (86), we need only show that

$$\sum_{i=1}^{k}(a_i - b_i) \geq 0, \quad k = 1, 2, \ldots, K-1. \tag{91}$$

Indeed the condition (14) is automatically satisfied here in the form $\sum_{i=1}^{K} a_i = \sum_{i=1}^{K} b_i = C + 1$ as a result of (88). The validity of (91) is now established by considering all possible cases.

Using (87), we readily see by direct inspection that (91) holds true whenever $1 \leq k < \min\{\alpha, \beta\}$ or $\max\{\alpha, \beta\} \leq k \leq K$, thus settling the case $\alpha = \beta$ as well. To complete the proof we need only show (91) for $\min\{\alpha, \beta\} \leq k < \max\{\alpha, \beta\}$. This is indeed the case when $\alpha < \beta$, for we then have

$$\sum_{i=1}^{k}(a_i - b_i) = 1 + \sum_{i=1}^{k}(n_i - m_i) \geq 1, \quad k = \alpha, \alpha+1, \ldots, \beta-1 \tag{92}$$

as a result of (87). On the other hand, if $\beta < \alpha$, we first note that

$$\sum_{i=1}^{k}(a_i - b_i) = -1 + \sum_{i=1}^{k}(n_i - m_i), \quad k = \beta, \beta+1, \ldots, \alpha-1 \tag{93}$$

and (91) follows if we show that $\sum_{i=1}^{k}(n_i - m_i) \geq 1$ on the range $\beta \leq k < \alpha$. With the inequalities (87) in mind, we need only show that each one of the equalities

$$\sum_{i=1}^{k}(n_i - m_i) = 0, \quad k = \beta, \beta+1, \ldots, \alpha-1 \tag{94}$$

leads to a contradiction: The equality (94) for $k = \beta$ is equivalent to $\beta(n_1 - m_1) = 0$, whence $n_1 = m_1 = \ldots = m_\beta$, and the inequality (91) for $k = \beta + 1$ easily yields $n_{\beta+1} - m_{\beta+1} = n_1 - m_{\beta+1} = m_\beta - m_{\beta+1} \geq 0$, a conclusion in contradiction with $m_\beta < m_{\beta+1}$ (since here $\beta < K$). On the other hand, if $\beta < k < \alpha$, then the equality (94) for such k can be rewritten as

$$n_1 = \frac{1}{k}\sum_{i=1}^{k} m_i \tag{95}$$

while (91) for $k+1$ again implies $n_{k+1} - m_{k+1} \geq 0$ (or equivalently, $m_{k+1} \leq n_1$), so that now we have $m_1 = m_2 = \ldots = m_\beta < m_{\beta+1} \leq \ldots \leq m_{k+1} \leq n_1$. The strict inequality in this chain of inequalities contradicts (95), and therefore (94) cannot hold. ■

8.3 Appendix C

We provide here a result that subsumes Lemma 11. First we introduce the definition of the strong stochastic ordering on the collection of $I\!R$-valued r.v.'s [37, 39]: Let X and Y be two such r.v.'s. We say that the r.v. X is stochastically smaller than the r.v. Y, and we write $X \leq_{st} Y$, if (15) holds for every increasing mapping $\phi : I\!R \rightarrow I\!R$, provided the expectations in (15) exist.

Lemma C.1. Consider an element $\mathbf{n}$ of $I\!N^K$ with $n_1 \leq n_2 \leq \ldots \leq n_K$, and assume that $b \geq 1$ tasks need to be scheduled. Given a non-empty subset I of $\{1, \ldots, K\}$, for each I-allocation $\mathbf{b}$ in $\mathcal{A}_I(b)$, there exists another $\{1, 2, \ldots, |I|\}$-allocation $\mathbf{b}'$ such that

$$\max_{1 \leq k \leq |I|} \{S^k_{n_k + b'_k}\} \leq_{st} \max_{k \in I} \{S^k_{n_k + b_k}\}. \tag{96}$$

Proof. With $|I| = L$, we have the representation $I = \{k_1, \ldots, k_L\}$ where $1 \leq k_1 < \ldots < k_L \leq K$. For any I-allocation $\mathbf{b}$ in $\mathcal{A}_I(b)$, we define the element $\mathbf{b}'$ of $I\!N^K$ as $b'_l = b_{k_l}$, $l = 1, \ldots, L$, and $b'_{L+1} = \ldots = b'_K = 0$. This vector $\mathbf{b}'$ is clearly an $\{1, \ldots, L\}$-allocation, and under the assumption on $\mathbf{n}$, we see that

$$n_l + b'_l = n_l + b_{k_l} \leq n_{k_l} + b_{k_l}, \quad l = 1, \ldots, L \tag{97}$$

since $l \leq k_l$ for all $l = 1, \ldots, L$. The enforced statistical assumptions readily imply $S^l_{n_l + b'_l} \leq_{st} S^{k_l}_{n_{k_l} + b_{k_l}}$, $l = 1, \ldots, L$, and (96) follows by a standard coupling argument ([37], Example 8.29a, p. 256). ■

References

[1] T. Axelrod, "Effects of synchronization barriers on multiprocessor performance," *Parallel Computing* **3** (1986), pp. 129–140.

[2] F. Baccelli and A. Makowski, "Queueing models for systems with synchronization constraints," *Proceedings of the IEEE* **77** (1989), pp. 138–161.

[3] F. Baccelli, A.M. Makowski and A. Shwartz, "The Fork–Join queue and related systems with synchronization constraints: Stochastic ordering and computable bounds," *Advances in Applied Probability* **21** (1989), pp. 629–660.

[4] F. Baccelli, W.A. Massey and D. Towsley, "Acyclic Fork–Join queueing networks," *J. Assoc. Comp. Mach.* **36** (1989), pp. 615–642.

[5] F. Baccelli and Z. Liu, "On the execution of parallel programs on multiprocessor systems: A queueing theory approach," *J. Assoc. Comp. Mach.* **37** (1990), pp. 371–413.

[6] P. Bilingsley, *Convergence of Probability Measures*, J. Wiley & Sons, New York (NY), 1968.

[7] C.S. Chang, "A new ordering for stochastic majorization: Theory and applications," *IBM Research Report* **RC–16028**, August 1990.

[8] K.L. Chung, *A Course in Probability Theory*, Second Edition, Academic Press, New York (NY), 1974.

[9] J.W. Cohen, *The Single Server Queue*, North–Holland, Amsterdam (The Netherlands), 1969.

[10] R. Cytron, "Useful parallelism in a multiprocessing environment," *Proceedings of the 1985 Conference on Parallel Processing.*

[11] M. Dubois, C. Scheurich and F. Briggs, "Synchronization, coherence, and event ordering in multiprocessors," *IEEE Transactions on Computers* **C–21** (1988), pp. 9–21.

[12] L. Durivault and R. Nelson, "An expression for the maximum of Erlang random variables," *IBM Research Report* **RC–16751**, January 1991.

[13] D.L. Eager, E.D. Lawzowska, and J. Zahorjan, "A comparison of receiver–initiated and sender–initiated adaptive load sharing," *Performance Evaluation* **6** (1986), pp. 53–68.

[14] "Adaptive load sharing in homogeneous distributed systems," *IEEE Transactions on Software Engineering* **SE–12** (1986), pp. 662–675.

[15] D.L. Eager, E.D. Lawzowska, and J. Zahorjan, "The limited performance benefits of migrating active processes for load sharing," *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 63–72, May 1988.

[16] L. Flatto and S. Hahn, "Two parallel queues created by arrivals with two demands I," *SIAM J. Appl. Math.* **44** (1984), pp. 1041–1053.

[17] H.P. Flatt and K. Kennedy, "Performance of parallel processors," *Parallel Computing* **12** (1989), pp. 1–20.

[18] A. Greenbaum, "Synchronization costs on multiprocessors," *Parallel Computing* **10** (1989), pp. 3–14.

[19] C. Kim and A.K. Agrawala, "Analysis of the Fork–Join queue," *IEEE Transactions on Computers* **C–38** (1989), pp. 250–255.

[20] L. Kleinrock, *Queueing Systems I: Theory*, J. Wiley & Sons, New York (NY), 1976.

[21] D.E. Knuth, *The Art of Computer Programming I: Fundamental Algorithms*, Second Edition, Addison–Wesley (1973).

[22] K. Lee and D. Towsley, "A Comparison of priority based decentralized load balancing policies," *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 70–77, May 1986.

[23] W. Leland and T. Ott, "Load balancing heuristics and process behavior," *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 54–69, May 1986.

[24] S. Leutenegger and M. Vernon, "The performance of multiprogrammed multiprocessor scheduling algorithms," *Performance Evaluation Review* **18** (1990), pp. 226–236.

[25] M. Litzkow, M. Livny and M. W. Mutka, "Condor – A hunter of idle workstations," *Proceedings of the 8th International Conference on Distributed Computing Systems*, San Jose (CA), June 1988.

[26] M. Livny and M. Melman, "Load balancing in homogeneous broadcast distributed systems," *Proceedings of the ACM Computer Network Performance Symposium*, pp. 47–55, 1982.

[27] S. Majumdar, D.L. Eager and R.B. Bunt, "Scheduling in multiprogrammed parallel systems," *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 104–113, 1988.

[28] A.M. Makowski and R. Nelson, "Distributed parallelism considered harmful," *IBM Research Report* RC 17448, 12/12/91, submitted to *IEEE Transactions on Distributed and Parallel Systems.*

[29] A.W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*, Academic Press, New York (NY), 1979.

[30] R. Mirchandaney, D. Towsley and J.A. Stankovic, "Analysis of the effects of delays on load sharing," *IEEE Transactions on Computers* **C–38** (1989), pp.1513–1525.

[31] R. Mirchandaney, D. Towsley and J.A. Stankovic, "Adaptive load sharing in heterogeneous systems," *Journal of Parallel and Distributed Computing*, pp. 331–346, 1990.

[32] R.F. Muirhead, "Some methods applicable to identities and inequalities of symmetric algebraic functions of n letters," *Proc. Edinburgh Math. Soc.* **21** (1903), pp. 144–157.

[33] A.J. Musciano and T. L. Sterling, "Efficient dynamic scheduling of medium-grained tasks for general purpose parallel processing," *Proceedings of 1988 International Conference on Parallel Processing* Vol. II, Software (H.E. Sturgis, Editor), pp. 166–175, 1988.

[34] R. Nelson, D. Towsley and A.N. Tantawi, "Performance analysis of parallel processing systems," *IEEE Transactions on Software Engineering* **SE–14** (1988), pp. 532-540.

[35] R. Nelson and A.N. Tantawi, "Approximate analysis of Fork/Join synchronization in parallel queues," *IEEE Transactions on Computers* **C–37** (1988), pp. 739–743.

[36] R. Nelson and T.K. Philips, "An approximation to the response time for shortest queue routing," *Performance Evaluation Review* **17** (1989), pp. 181–189.

[37] S. Ross, *Stochastic Processes*, J. Wiley & Sons, New York (NY), 1984.

[38] K. Sevcik, "Characterizations of parallelism in applications and their use in scheduling," *Performance Evaluation Review* **17** (1989), pp. 171–180.

[39] D. Stoyan, *Comparison Methods for Queues and Other Stochastic Models*, English Translation (D.J. Daley, Editor), J. Wiley & Sons, New York (NY), 1984.

[40] M. Squillante and R. Nelson, "Analysis of task migration in shared-memory multiprocessor scheduling," *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 143–155, May 1991.

[41] J. Zahorjan and C. McCann, "Processor scheduling in shared memory multiprocessors," *Performance Evaluation Review* 18 (1990), pp. 214–225.

Figure 1. The Model

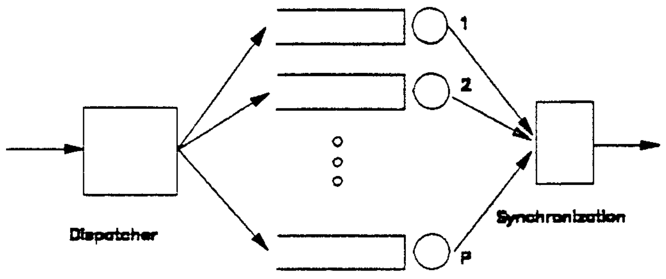


Figure 2. K=2, b=2, exponential service time

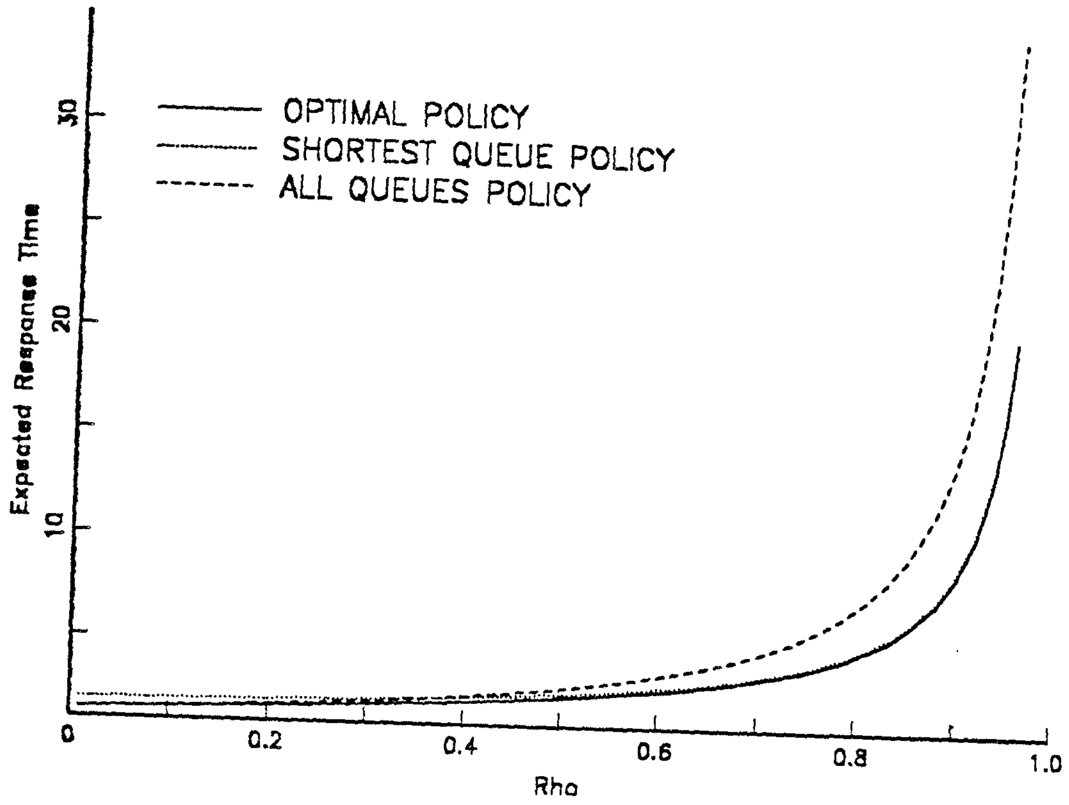


Figure 3. K=2, b=2, exponential service time

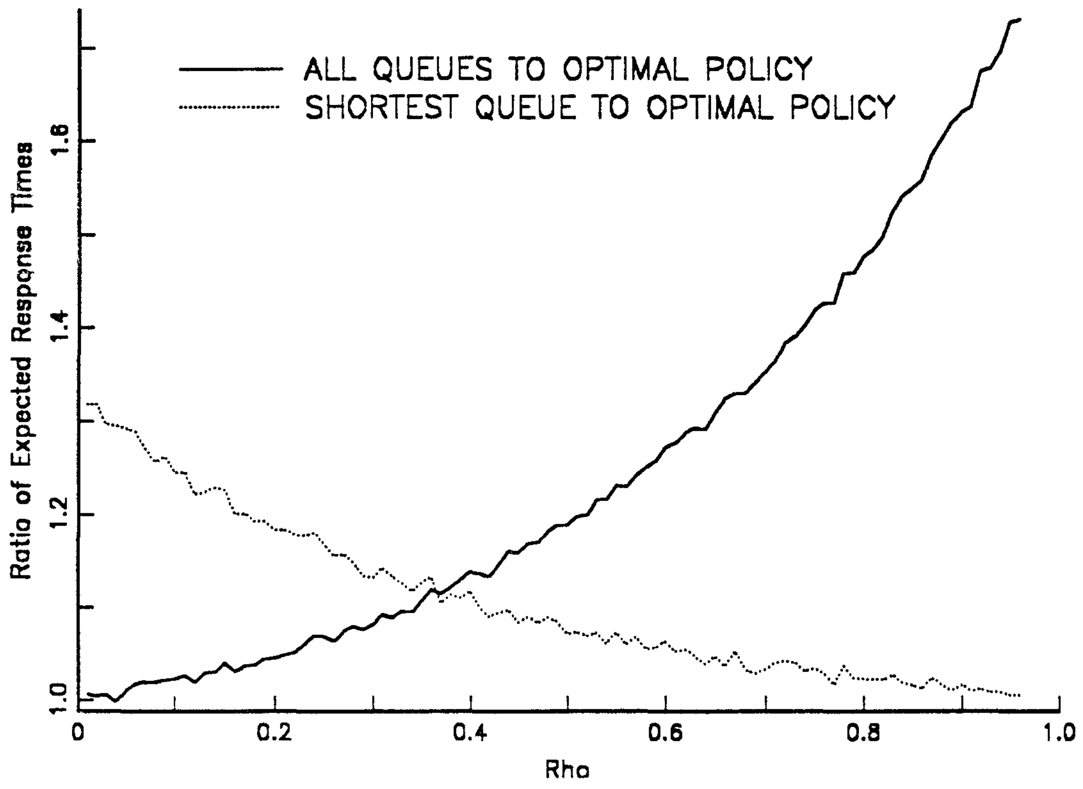


Figure 4. K=2, b=2, exponential service time

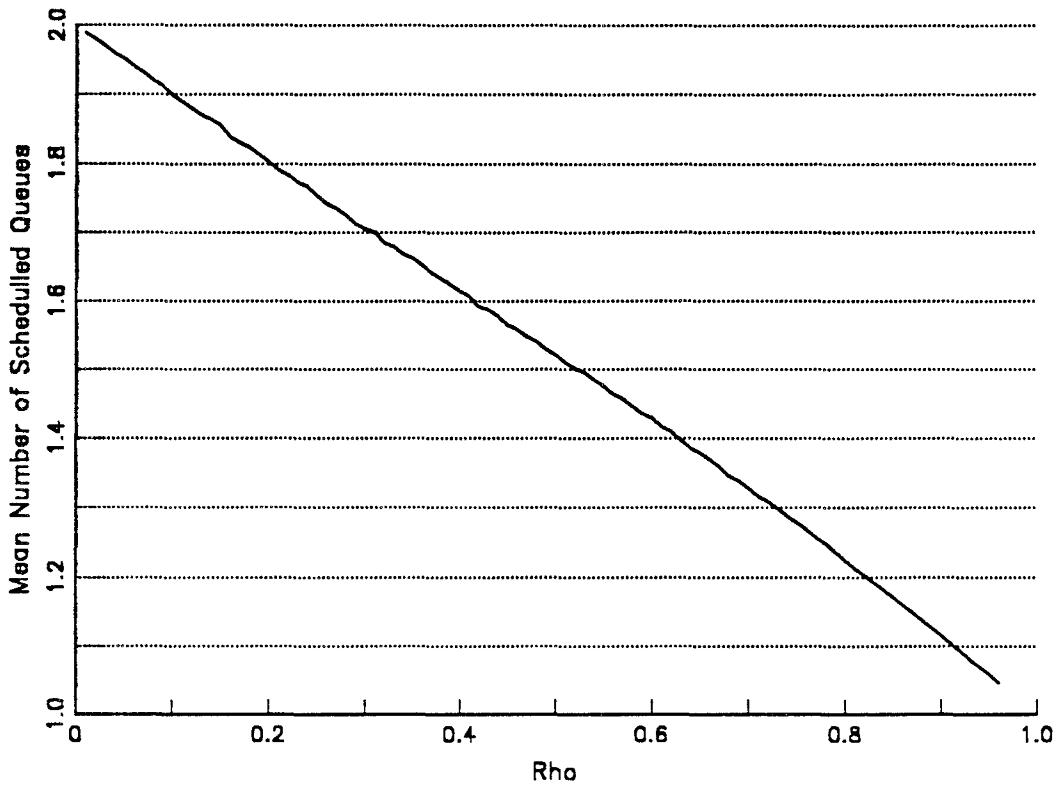


Figure 5. K=4, b=8, exponential service time

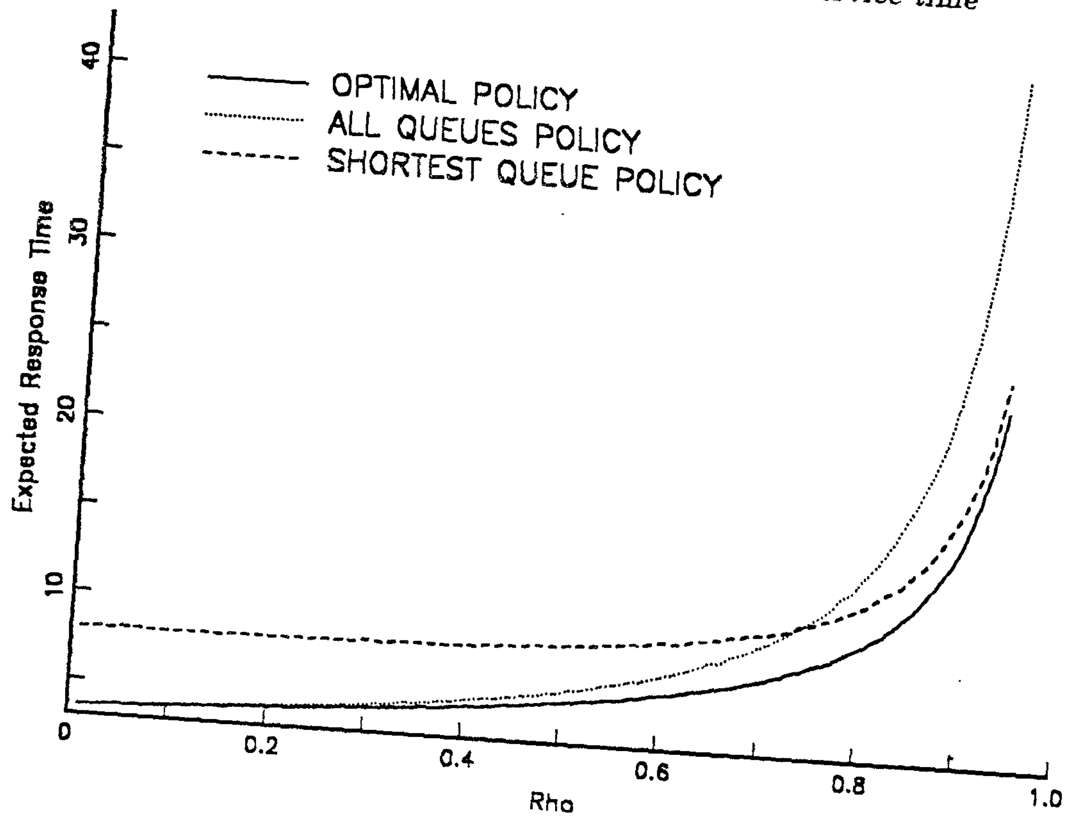


Figure 6. K=4, b=8, exponential service time

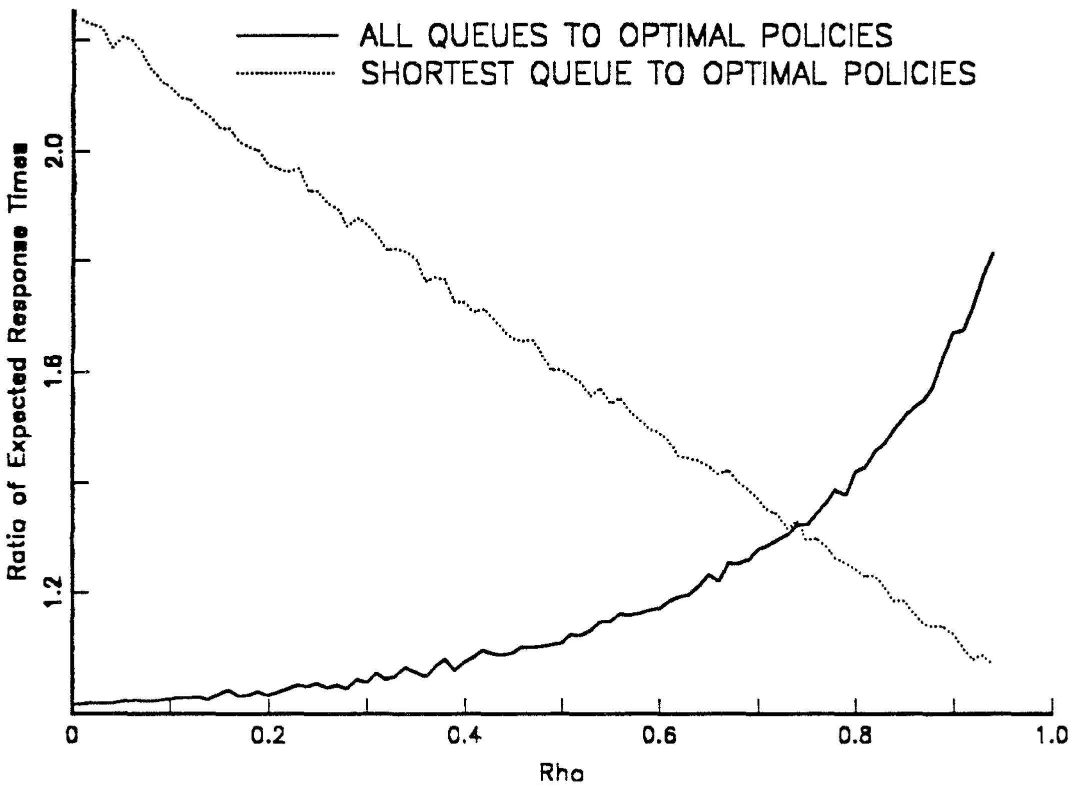


Figure 7. K=4, b=8, exponential service time

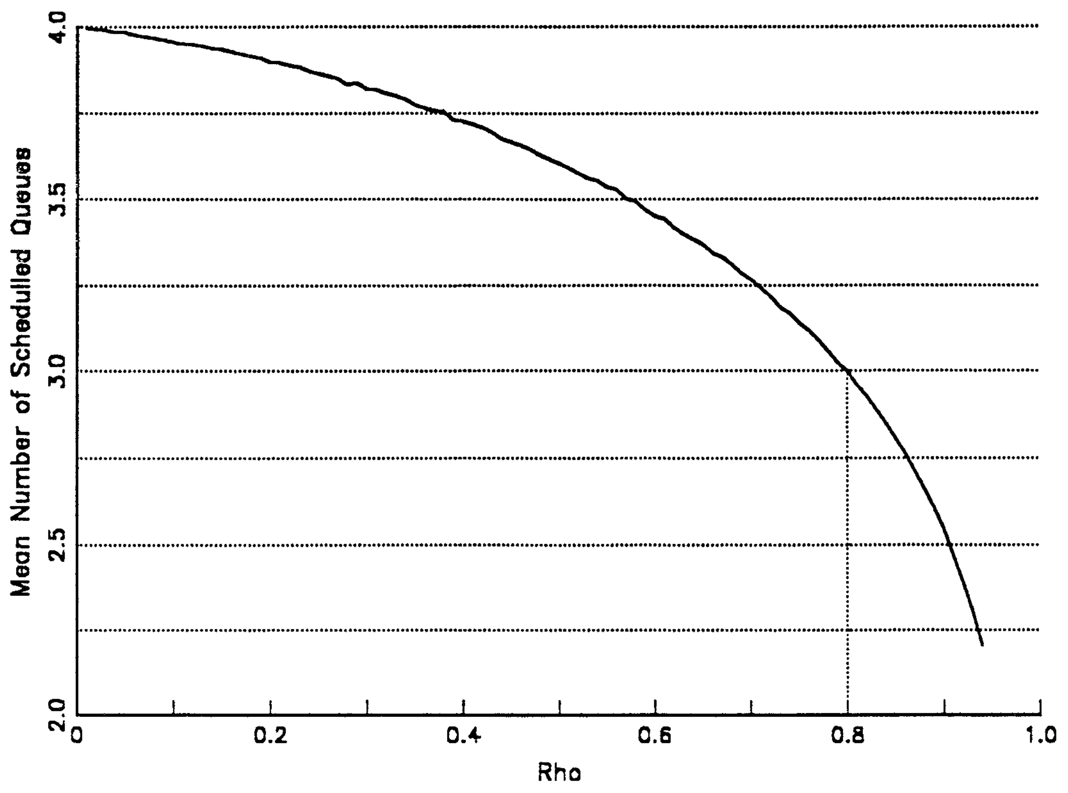


Figure 8. K=4, b=4, optimal policy

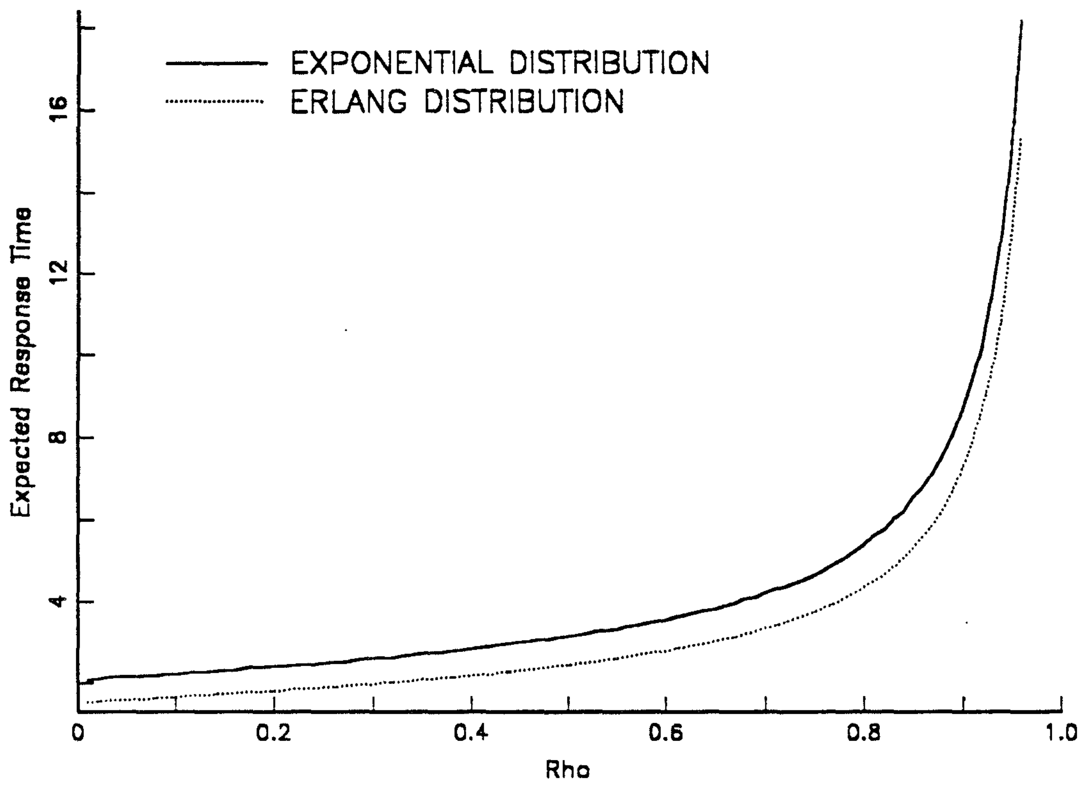


Figure 9. K=4, b=4

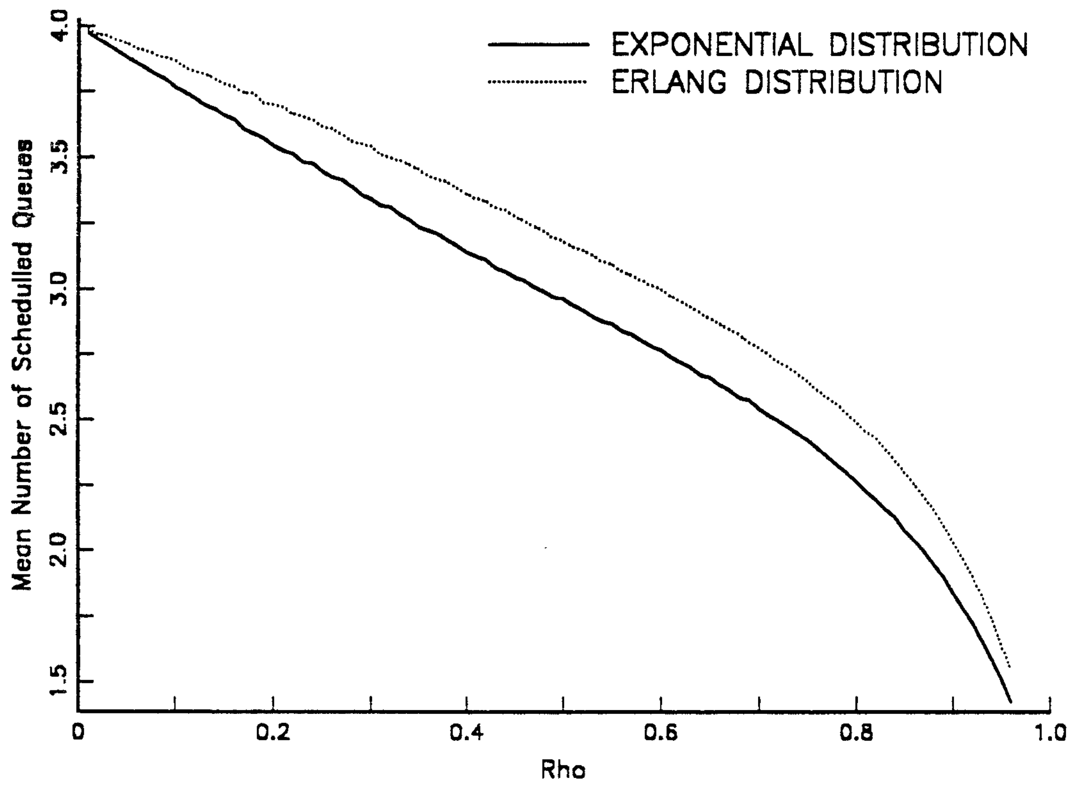


Figure 10. K=4, b=4, Erlang distribution

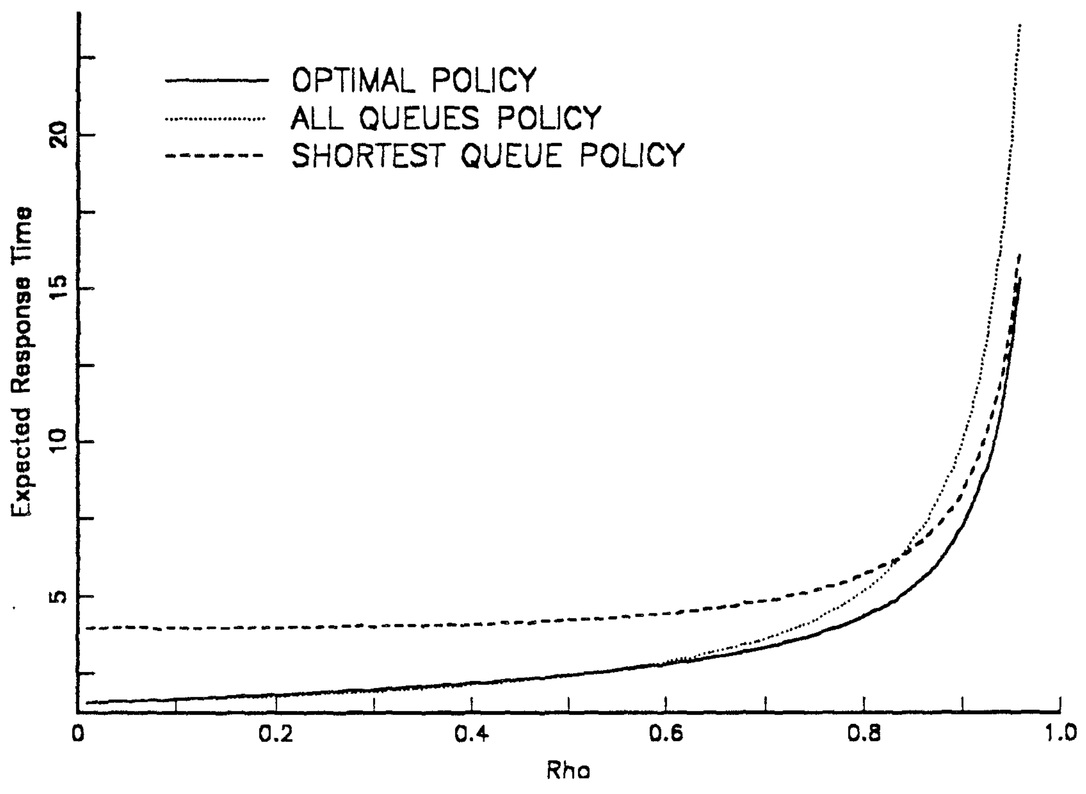


Figure 11. K=4, b=4, Erlang distribution

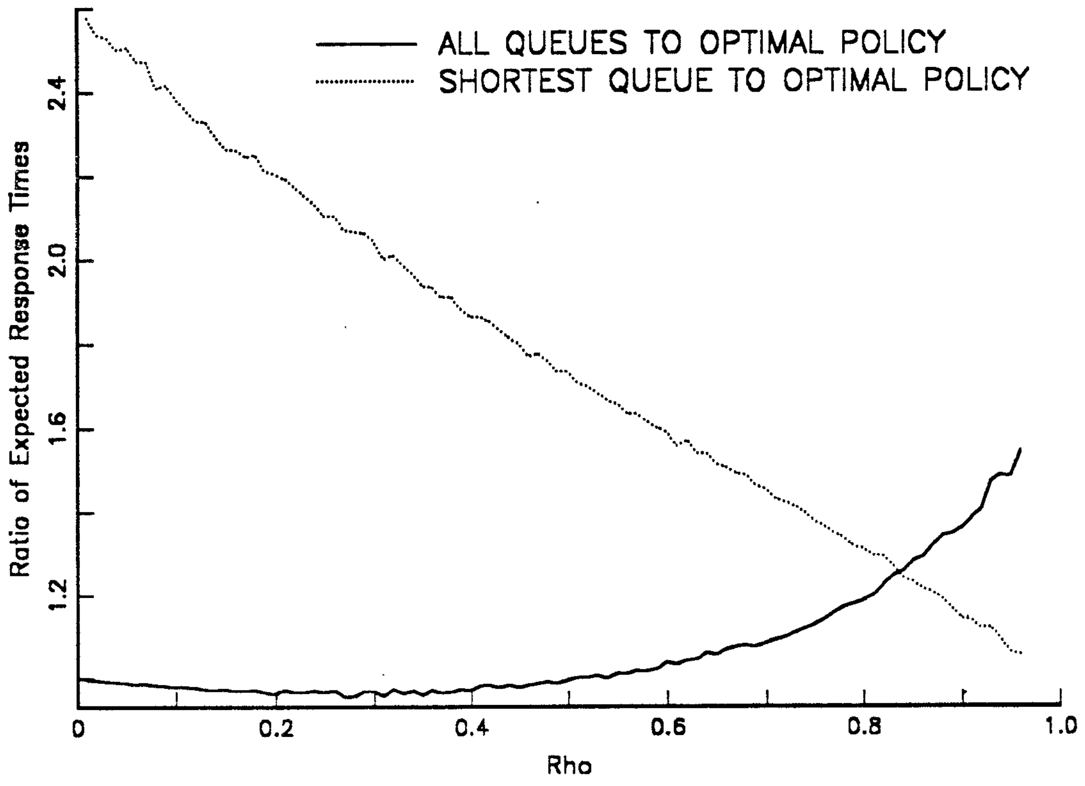


Figure 12. Ratios optimal policy to speed-up

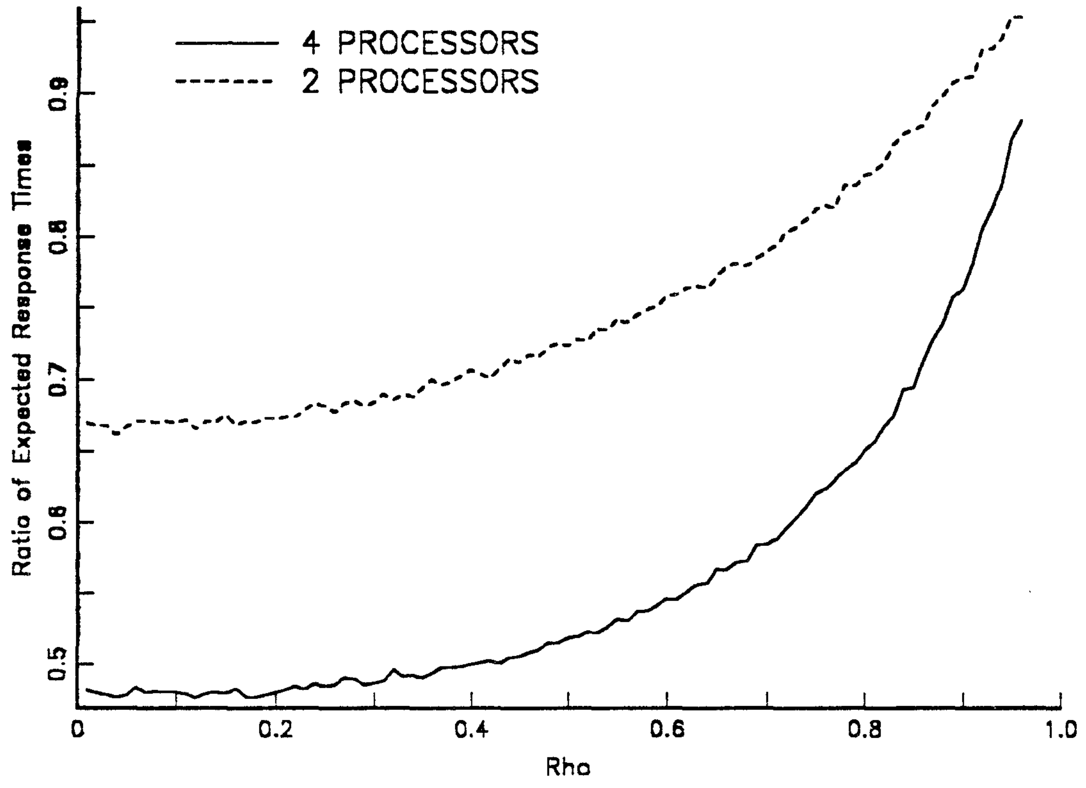
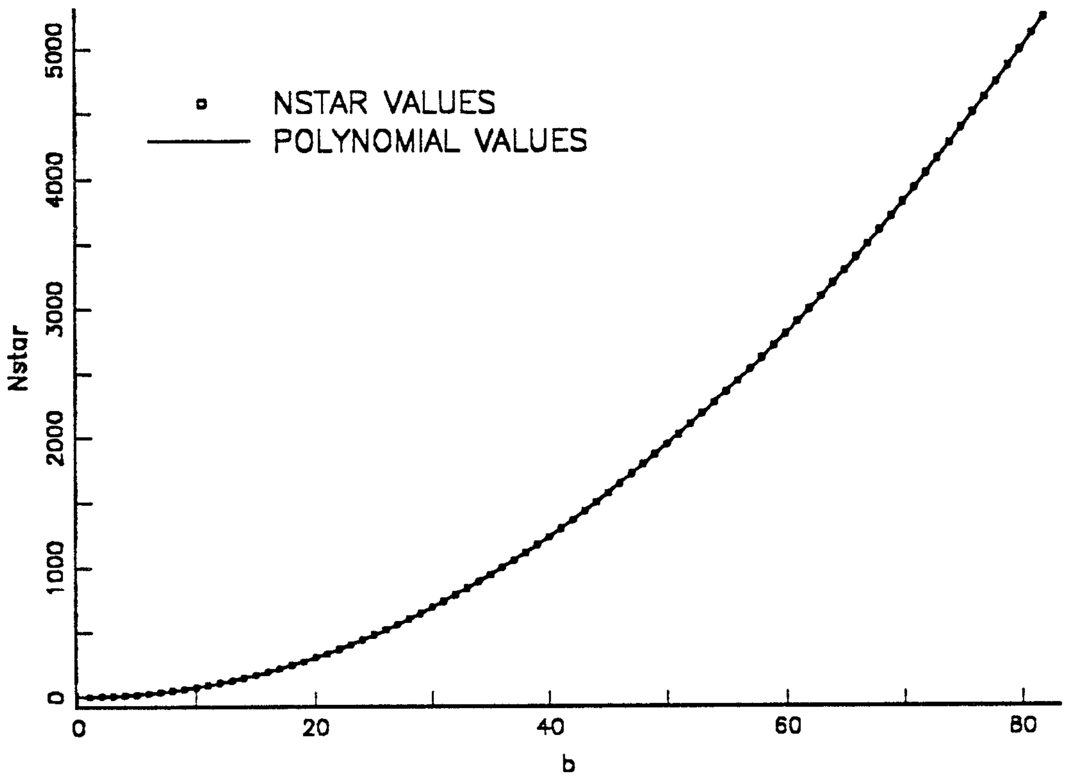


Figure 13. n^* as a function of b for K=2


NSTAR VALUES
POLYNOMIAL VALUES
Nstar
b