Optimal Scheduling of Reader-Writer Systems

K. D. Glazebrook
Department of Statistics,
University of Newcastle upon Tyne,
NE1 7RU, England

V. G. Kulkarni
Department of Operations Research
University of North Carolina
Chapel Hill, N.C. 27599-3180

L. C. Puryear, Jr.,
303 Heidinger Dr.,
Cary, NC 27511

August 29, 2000

Abstract

We consider a reader-writer system consisting of a single server and a fixed number of jobs (or customers) belonging to two classes. Class one jobs are called readers and any number of them can be processed simultaneously. Class two jobs are called writers and they have to be processed one at a time. When a writer is being processed no other writer or readers can be processed.

A fixed number of readers and writers are ready for processing at time 0. Their processing times are independent random variables. Each reader and writer has a fixed waiting cost rate. We find optimal scheduling rules that minimize the expected total waiting cost (expected total weighted flowtime). We consider both nonpreemptive and preemptive scheduling. The optimal nonpreemptive schedule is derived by a variation of the usual interchange argument, while the optimal schedule in the preemptive case is given by a Gittins index policy. These index policies continue to be optimal for systems in which new writers enter the system in a Poisson fashion.
1 Introduction

We consider a reader-writer system consisting of a single server and a fixed number of jobs (or customers) belonging to two classes. Class one jobs are called readers and any number of them can be processed simultaneously. Class two jobs are called writers and they have to be processed one at a time. When a writer is being processed no other writer or readers can be processed.

Such processes arise as simple models of database systems: the readers of the database do not change the database and hence many readers can be allowed to utilize service simultaneously. This number is typically limited by the number of replications of the database, or by the number of processors that can simultaneously access the shared memory. Writers, on the other hand, change the database, and hence, to preserve data integrity, only one writer can be given access to the database at a time, and all other readers and writers are blocked. Note that this simple description ignores other important features of the database systems, e.g., granularity, locking mechanisms, sequential queries, etc. See Ullman [12] for details. The main question is how to schedule the readers and writers so as to optimize system performance.

There is a significant literature analysing the performance of specific scheduling disciplines in reader-writer systems. Puryear [9] studies five priority scheduling disciplines, ranging from preemptive priority to readers to preemptive priority to writers. (Also see Kulkarni and Puryear [6, 7, 8, 10].) Reiman and Wright [11] study the first come first served discipline.

Little work is done which aims to determine optimal scheduling policies for reader-writer systems. Courcoubetis and Reiman [1] consider a reader-writer queue with exponential processing times and multiple servers. The readers are always available for service, while writers arrive according to a Poisson process. They derive optimal scheduling policies for the writers.

As far as the authors know, there are no results concerning optimal scheduling policies for the standard reader-writer queue as described in the first paragraph of this section when external arrivals of readers and writers are allowed. Our own experience suggests that this is a very difficult problem. In this paper we consider primarily a simple version with no arrivals. This assumption brings the problem into the realm of standard scheduling theory. Although there are results on optimal scheduling under batch processing when processing times are deterministic (see Webster and Baker [14] and references therein), none of the models address the reader-writer system situation. We are in fact able to analyse the reader-writer problem in which new writers (only) are allowed to arrive in a Poisson fashion, and preemptions are allowed without penalty. This is closely comparable to the nonpreemptive version of Courcoubetis and Reiman [1].

We describe the model in the next section. Section 3 contains an analysis of nonpreemptive policies while the preemptive case is considered in Section 4. In the nonpreemptive case, the key issue is how to batch the readers and in what sequence to process these batches and the writers. In the preemptive case, it is clearly optimal always to process all the readers together whenever it is optimal to process the readers at all. Hence, the problem is in a way easier: we only have to identify when it is optimal to process the readers. Consequently, we have more complete results for the preemptive case in comparison to the nonpreemptive case. Possible extensions to this work are mentioned in the concluding section.
2 The Model

Consider a reader-writer system with $m$ readers and $n$ writers present at time 0. There are no arrivals after time 0. Let $R_i$ be the random processing time of reader $i$ ($1 \leq i \leq m$), with $r_i = E(R_i)$. Similarly, $W_j$ is the processing time of writer $j$ ($1 \leq j \leq n$), and $w_j = E(W_j)$. We assume that all the processing times are independent.

It costs $c_i$ to hold reader $i$ in the system for one unit of time, and $d_j$ to keep writer $j$ in the system for unit of time. The readers and writers leave the system as soon as their processing is finished. The aim is to find a processing schedule to minimize the expected total holding cost of the readers and writers. This is equivalent to minimizing the expected total weighted flow-time.

We consider two types of schedule: (i) nonpreemptive schedules, under which once a job starts service it is processed until completion without interruption; and (ii) preemptive schedules, under which a job can be interrupted any number of times, without penalty.

As an example, suppose there are two readers (labeled $r1$ and $r2$) and two writers (labeled $w1$ and $w2$). Using the notation || to mean “process in parallel”, we enumerate three nonpreemptive schedules and the corresponding expected costs:

1. Schedule=$w1, w2, r1, r2$.
   Expected cost = $d_1 \ast w_1 + d_2 \ast (w_1 + w_2) + c_1 \ast (w_1 + w_2 + r_1) + c_2 \ast (w_1 + w_2 + r_1 + r_2)$.

2. Schedule=$w1, w2, (r1||r2)$.
   Expected cost=$d_1 \ast w_1 + d_2 \ast (w_1 + w_2) + c_1 \ast (w_1 + w_2 + r_1) + c_2 \ast (w_1 + w_2 + r_2)$.

3. Schedule=$(r1||r2), w1, w2$.
   Expected cost=$c_1 \ast r_1 + c_2 \ast r_2 + d_1 \ast (E(\max(R_1, R_2)) + w_1) + d_2 \ast (E(\max(R_1, R_2)) + w_1 + w_2)$.

Note that in schedule 3, writer $w1$ waits for $\max(R_1, R_2)$ before starting its processing. This is because readers $r1$ and $r2$ are being processed in parallel, and preemption is not allowed.

3 Optimal Nonpreemptive Schedules

In this section, we consider nonpreemptive schedules in which a reader or a writer cannot be preempted once its processing starts. In general, a nonpreemptive policy can allow the history of the process (e.g., past processing times) to influence the future choices. On the other hand, a nonpreemptive policy may be static, i.e., it is given as an immutable ordering of processing in advance of actual processing. Such policies are sometimes called permutation policies. Of course, it may be true that a permutation policy is optimal in the class of all nonpreemptive policies.

Although the restriction to nonpreemptive schedules allows one to start processing a new reader during the processing of a batch of readers, a simple sample path argument shows that an optimal policy would not resort to such schedules. Then, the assumptions about the linear holding costs and independent processing times imply that an optimal nonpreemptive policy must be a permutation policy of a kind described below.
Let \( \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k \) be a partition of \( \{1, 2, \ldots, m\} \). The set of readers \( \mathcal{R}_i \) is called the \( i^{th} \) super-reader. Now consider the problem of determining an optimal schedule in which the \( k \) super-readers (with the \( i^{th} \) super-reader having processing time \( \max\{R_j : j \in \mathcal{R}_i\} \)) and the \( n \) writers must be processed one at a time. Note that we do not allow the processing of two or more super-readers simultaneously. For this restricted problem the standard interchange argument can be used to derive the optimal scheduling policy.

To derive the optimal policy we construct a single machine job-shop with \( n + k \) jobs as follows: writer \( i \) (\( 1 \leq i \leq n \)) is the \( i^{th} \) job with mean processing time \( w_i/d_i \) (or an index \( d_i/w_i \)), and the \( i^{th} \) super-reader (\( 1 \leq i \leq k \)) is the \((n+i)^{th}\) job with mean processing time

\[
w_{n+i} = \frac{E(\max\{R_j : j \in \mathcal{R}_i\})}{\sum_{j \in \mathcal{R}_i} c_j}
\]  

(1 \leq i \leq k) (or an index \( \frac{\sum_{j \in \mathcal{R}_i} c_j}{E(\max\{R_j : j \in \mathcal{R}_i\})} \)). The optimal policy is given in the following theorem.

**Theorem 1** The optimal policy processes the \( n + k \) jobs \( (n \) writers and the \( k \) super-readers) according to the Shortest Mean Processing Time First (i.e., Smith’s rule, or the highest index first).

**Proof:** Follows by standard interchange arguments. ♠

Thus the optimal processing schedule for the \( n \) writers and the given super-readers \( \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k\} \) is given by Theorem 1. The next problem is to find an optimal partition of readers into super-readers. Unfortunately, this is not an easy problem in the general case. However, the above theorem does offer a valuable insight stated in the following theorem:

**Theorem 2** If the optimal schedule produced by Theorem 1 has two super-readers adjacent to each other, then the reader partition in which these two super-readers are combined into a single super-reader is at least as good.

**Proof:** Suppose \( \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_k\} \) is a given reader partition. Without loss of generality assume that the optimal schedule of the \( n + k \) jobs (call it \( S^* \)) produced by Theorem 1 puts super-reader \( \mathcal{R}_1 \) in the \( i^{th} \) position and \( \mathcal{R}_2 \) in the \((t + 1)^{st} \) position. Suppose the sum of the cost rates of the readers and writers that are processed after \( \mathcal{R}_2 \) is \( b \).

Now consider a reader partition with \( k - 1 \) subsets \( \{\mathcal{R}_1 \cup \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_k\} \) and consider a new schedule of the \( n + k - 1 \) jobs (call it \( S \)) that is the same as \( S^* \) except that the super-reader \( \mathcal{R}_1 \cup \mathcal{R}_2 \) is in position \( t \). Thus the job in position \( l \) in \( S^* \) is processed in position \( l \) in \( S \) if \( l < t \), and in position \( l - 1 \) in \( S \) if \( l > t + 1 \). Then we have

\[
E(\text{Cost}(S^*)) - E(\text{Cost}(S)) = \left( \sum_{j \in \mathcal{R}_2} c_j \right) E(\max\{R_j : j \in \mathcal{R}_1\})
\]

\[+ b * (E(\max\{R_l : l \in \mathcal{R}_1\}) + E(\max\{R_l : l \in \mathcal{R}_2\}) - E(\max\{R_l : l \in \mathcal{R}_1 \cup \mathcal{R}_2\})).
\]

Since

\[E(\max\{R_l : l \in \mathcal{R}_1 \cup \mathcal{R}_2\}) \leq E(\max\{R_l : l \in \mathcal{R}_1\}) + E(\max\{R_l : l \in \mathcal{R}_2\})\]

it is clear that schedule \( S \) is at least as good as \( S^* \). Hence the optimal schedule for the new partition is at least as good as \( S^* \). Hence the new partition is at least as good as the old one. ♠
**Special Case:** Now we consider the following special case:

(i) Reader processing times are iid with common distribution $R(\cdot)$, common mean $r$, and common holding cost rate $c$.

(ii) Writer processing times are independent with $j^{th}$ writer having distribution $W_j(\cdot)$, mean $w_j$ and holding cost rate $d_j$. We shall assume without loss of generality that $\frac{w_j}{d_j}$ is a nondecreasing function of $i$.

We shall first prove that all the readers will be processed as a single super-reader in an optimal policy under the above assumptions. Let $S_i$ be the following schedule: process $i$ readers first in parallel, then process all the writers one at a time in increasing order of $w_j/d_j$, and finally process all the remaining $m - i$ readers in parallel. Notice we allow $i = 0$ (process all writers first) and $i = m$ (process all readers first).

Now let $f_0 = 0$ and for $1 \leq i \leq m$ let

$$f_i = E(\max\{R_1, R_2, ..., R_i\})$$

(2)

where $R_1, R_2, ..., R_m$ are iid random variables each with distribution $R(\cdot)$. The following lemma is a simple consequence of the convexity of $\{f_i, 0 \leq i \leq m\}$.

**Lemma 1.** $f_i/i, i \geq 1$ is a decreasing function of $i$.

We shall find the following notation useful:

$$w = \sum_{j=1}^{n} \frac{w_j}{n},$$

(3)

$$d = \sum_{j=1}^{n} \frac{d_j}{n}.$$  

(4)

**Theorem 3** Assume (i) and (ii). Among the schedules $\{S_i, 0 \leq i \leq m\}$, schedule $S_0$ (i.e., process all writers first) is optimal if

$$\frac{w}{d} \leq \frac{1}{c} \frac{f_m}{m}.$$  

Otherwise schedule $S_m$ (i.e., process all readers first) is optimal.

**Proof:** A simple interchange argument shows that the schedule that processes all writers first and then the readers in two batches of $i$ and $m - i$ readers is better than $S_i$ if

$$\frac{w}{d} \leq \frac{1}{c} \frac{f_i}{i}.$$  

Furthermore, it follows from theorem 2 that it is better to process all the readers together, and hence (under the above condition) schedule $S_0$ is at least as good as $S_i$. By a similar argument, $S_m$ is at least as good as $S_i$ if

$$\frac{w}{d} \geq \frac{1}{c} \frac{f_{m-i}}{m - i}.$$  

Now consider the following cases:
1. \( \frac{w}{d} \leq \frac{f_m}{c/m} \).

It follows from the above argument and Lemma 1 that in this case the schedule \( S_0 \) (i.e. process all writers first) is optimal among all \( \{S_i, 0 \leq i \leq m\} \).

2. \( \frac{w}{d} \geq \frac{1}{c} \frac{f_1}{t} \).

By a similar argument it follows in this case the schedule \( S_m \) (i.e. process all readers first) is optimal among all \( \{S_i, 0 \leq i \leq m\} \).

3. \( \frac{f_{k+1}}{c/k} \leq \frac{w}{d} \leq \frac{f_k}{c/k} \) for some \( k \in \{1, 2, \ldots, m-1\} \).

Note that schedule \( S_i \) is a candidate for an optimal schedule (among \( S_0, \ldots, S_m \)) iff

\[
\frac{1}{c} \frac{f_i}{i} \leq \frac{w}{d} \leq \frac{1}{c} \frac{f_{m-i}}{m-i}.
\]

This along with the definition of \( k \) above implies that \( \max\{k+1, m-k\} \leq i \leq m \). Now, the expected cost of the \( i^{th} \) schedule is given by

\[
C(i) = f_i (nd + (m-i)c) + nw(m-i)c + A,
\]

where \( A \) is independent of \( i \). Seeking to minimize this quantity over \( \max\{k+1, m-k\} \leq i \leq m \), we compute

\[
C(i) - C(i-1) = f_i (nd + (m-i)c) + nw(m-i)c
- [f_{i-1} (nd + (m-i+1)c) + nw(m-i+1)c]
= (nd + (m-i+1)c)(f_i - f_{i-1}) - cf_i - nwc
= (nd + (m-i+1)c)(f_i - f_{i-1})/i - cf_i - nwc
\leq (nd + (m-i+1)c)f_i/i - cf_i - nwc
\text{since } f_i(f_i - f_{i-1}) \leq f_i \text{ from Lemma 1.}
= (nd + (m-2i+1)c)f_i/i - nwc
\leq (nd + (m-m-1+1)c)f_i/i - nwc \text{ since } 2i \geq m+1
= ncd(\frac{1}{c} \frac{f_i}{i} - \frac{w}{d})
\leq 0
\]

where the last inequality follows from the assumption of this case. This shows that \( C(i), \max\{k+1, m-k\} \leq i \leq m \) is minimized at \( i = m \). Thus it is optimal to process all the readers first.

4. \( m \) is even, \( \frac{f_{m/2}}{c/m/2} = \frac{w}{d} \).

In this case schedule \( i \) is a candidate if \( m/2 \leq i \leq m \). The rest of the analysis is identical to case 3, and it follows that schedule \( m \) is optimal.
This completes the proof. ♣

Using the above result we shall show the following

**Theorem 4** All the readers must be in a single block in an optimal policy.

**Proof:** Suppose not. Then we can write the optimal schedule as

\[ A \mathcal{R}_1(w_j w_{j+1} \ldots w_k) \mathcal{R}_2 B \]

where the last member of \( A \) and the first member of \( B \) are both writers, and \( \mathcal{R}_1 \) and \( \mathcal{R}_2 \) are two super-readers. By Theorem 3 (with \( m = k - j + 1 \)) and Theorem 2 it is easy to see that either

\[ A(\mathcal{R}_1 \cup \mathcal{R}_2)(w_j w_{j+1} \ldots w_k)B \]

or

\[ A(w_j w_{j+1} \ldots w_k)(\mathcal{R}_1 \cup \mathcal{R}_2)B \]

must be at least as good as the original schedule. Proceeding in this fashion we see that all readers can eventually be processed as a single block without increasing the cost. This proves the theorem.

♣

Using the above theorem we get the following

**Theorem 5** Assume (i) and (ii), and suppose there is an integer \( 0 \leq r \leq n \) such that

\[ \frac{w_1}{d_1} \leq \frac{w_2}{d_2} \leq \ldots \leq \frac{w_r}{d_r} \leq \frac{f_m}{mc} \leq \frac{w_{r+1}}{d_{r+1}} \leq \ldots \leq \frac{w_n}{d_n}. \]

Then the optimal schedule processes writers 1, 2, \ldots, \( r \) one at a time in that order, followed by all the readers simultaneously, followed by writers \( r+1, r+2, \ldots, n \) one at a time in that order.

Note that the case \( \frac{f_m}{mc} < \frac{w_1}{d_1} \) implies \( r = 0 \), i.e. it is optimal to process all readers first. Similarly, the case \( \frac{f_m}{mc} > \frac{w_n}{d_n} \) implies that it is optimal to process all the writers first. We illustrate the above result with an example.

**Example 1:** Suppose the processing time of the \( i \)th reader is exponentially distributed with mean \( r \), while that of the \( j \)th writer is exponentially distributed with mean \( w \). Furthermore assume that \( c_i = c \) for all \( 1 \leq i \leq m \) and \( d_j = d \) for all \( 1 \leq j \leq n \). Then the optimal schedule nonpreemptively processes all the readers simultaneously before any writers if

\[ \frac{r}{mc} \sum_{i=1}^{m} \frac{1}{i} < \frac{w}{d}. \]  \( (7) \)

Otherwise all the writers are processed one by one before any readers, and all the readers are simultaneously processed at the end.

**Remark:** Relaxing assumption (i) further seems difficult in the nonpreemptive case. For example, if the mean processing times of the readers are different, then it may be advantageous to process the short readers simultaneously first, and the long readers simultaneously last. It is also difficult to allow external arrivals of readers or writers. We shall see in the next section that assumptions (i) and (ii) can be removed in the preemptive case.
4 Optimal Preemptive Schedules

Here we shall consider preemptive schedules in which at all decision epochs the system controller may choose to allocate processing to any individual writer still in the system or to some of the remaining readers. The problem is hugely simplified by the observation that an elementary sample path argument shows that an optimal schedule will always process all readers whenever it processes any. For simplicity, we shall assume that time proceeds in discrete steps \( t = 0, 1, 2, \ldots \) and that the processing times \( R_t \) and \( W_j \) are positive integer-valued random variables. The number of writers in the system at \( t = 0 \) is \( n \).

The problem can be modelled as an undiscounted multi-armed bandit in which a decision-maker must choose between \( n + 1 \) options at each decision epoch \( t = 0, 1, 2, \ldots \). Of these options, \( n \) represents a choice of one of the writers for processing, while the \( (n + 1) \)st is the choice of granting processing to all remaining readers. By a result due to Walrand [13], optimal policies are determined by giving to each option a calibrating index (herein referred to as a Gittins index), which is a function of its current state. At each decision epoch an optimal policy chooses one of the options with largest current index. It does not matter how ties are broken. Hence the problem reduces to that of obtaining the indices.

The index for an option has a simple interpretation as the maximal rate at which total holding costs for the system can be driven down by choosing that option for some time period. Determining the Gittins index for each of the writers is routine and we shall not give the details. If writer \( j \) has received \( x \) units of processing to date and has yet to complete, its index is given by

\[
G_j(x) = d_j \sup_{r \in \mathbb{Z}^+} \frac{\sum_{s=0}^{r-1} P(W_j = x + s \mid W_j > x) \sum_{s=0}^{r-1} P(W_j > x + s \mid W_j > x)}{\sum_{s=0}^{r-1} P(W_j > x + s \mid W_j > x)}
\]

If a writer’s processing is complete, the associated index is zero.

Obtaining the index for the reader process is much more complex. To develop the index, suppose that the reader population is in some general state \((M, y)\) at time 0: namely, there are \( M \) readers still in the system (i.e. their processing is not yet complete) at 0 and \( y \) is their common elapsed processing time. We suppose now that processing is allocated to the readers at times \( t = 0, 1, \ldots, \tau - 1 \) for some positive stopping time \( \tau \). \( \{M(t), Y(t)\} \) denotes the state of this reader population at time \( t \). We plainly must have \( M(t) \leq M \) and \( Y(t) = y + t \). The corresponding holding cost rate for the readers is denoted \( c\{M(t), Y(t)\} \), where

\[
c\{M(t), Y(t)\} = \sum_{i \in M(t)} c_i.
\]

The Gittins index for the reader population in state \((M, y)\) is given by

\[
G(M, y) = \sup_{\tau} G\{(M, y) ; \tau\},
\]

where

\[
G\{(M, y) ; \tau\} = (c\{M, y\} - E[c\{M(\tau), Y(\tau)\}])/E(\tau).
\]

The difficulty in obtaining the index in (10) is rooted in the complexity of the class of stopping times over which the supremum is taken. Following Gittins [3, Chapter 4], progress can be made
in cases for which the indices are monotone. Such index properties turn out to be related to monotonicity properties of the completion rate (or hazard rate) functions $\Psi_i$, defined by

$$
\Psi_i(x) = P(R_i = x + 1 \mid R_i > x) = \{R_i(x + 1) - R_i(x)\} \{1 - R_i(x)\}^{-1},
$$

(12)

where $R_i(\cdot)$ is the distribution function of $R_i$.

**Theorem 6** If $R_i \sim DHR$, i.e. $\Psi_i$, $1 \leq i \leq m$, is nonincreasing, then

$$
G(M, y) = \sum_{i \in M} c_i \Psi_i(y)
$$

(13)

**Proof:** Let $\tau_1$ denote the stopping time equal to 1 almost surely. From (11), it is straightforward that

$$
G\{M, y; \tau_1\} = \sum_{i \in M} c_i P(R_i = y + 1 \mid R_i > y) = \sum_{i \in M} c_i \Psi_i(y).
$$

(14)

It then follows simply from the hypothesis of the theorem that the stochastic process

$$
G\{M(t), Y(t); \tau_1\} = \sum_{i \in M(t)} c_i \Psi_i\{Y(t)\}
$$

(15)

is nonincreasing in $t$ almost surely. This is an example of the deteriorating case discussed in Gittins [ [3], Chapter 4]. It is a simple matter to show that in this case the supremum in (10) is attained at $\tau = \tau_1$. The result follows from (14).

\[ \blacklozenge \]

That there can be no comparable case in which the Gittins index increases almost surely throughout the evolution of the reader process is reasonably clear from the fact that the instantaneous rate of cost reduction at $t$, viz., $\sum_{i \in M(t)} c_i \Psi_i\{Y(t)\}$, will decrease upon completion of any reader’s requirement. The appropriate result to aim for in the case in which the residual processing requirements of readers decrease stochastically as elapsed processing increases is that the Gittins index for the reader process should increase between successive times at which readers leave the system. This is expressed in Theorem 8. Before proceeding to that, we require the following monotonicity property, which is of independent interest.

**Theorem 7**

$$
M_1 \subseteq M_2 \Rightarrow G(M_1, y) \leq G(M_2, y), y \in [0, \infty).
$$

**Proof:** Let $\hat{\tau}$ be a positive stopping time which realizes $G(M_1, y)$ - i.e. such that

$$
G(M_1, y) = G\{(M_1, y); \hat{\tau}\}
$$

(16)

Now consider the reader process in initial state $(M_2, y)$ where $M_1 \subseteq M_2$. Consider also the allocation of processing to the readers in $M_2$ up to stopping time $\hat{\tau}$, where $\hat{\tau}$ is obtained by restricting attention to the readers in subset $M_1$ and stopping at $\hat{\tau}$, defined with respect to the evolution of the $M_1$-readers only. By the independence of the $R_i$, $\hat{\tau}$ is stochastically identical to $\hat{\tau}$. By (11)

$$
G\{(M_2, y); \hat{\tau}\}E(\hat{\tau}) = G\{(M_1, y), \hat{\tau}\}E(\hat{\tau}) + \sum_{i \in M_2 \setminus M_1} c_i [1 - E\{I_i(\hat{\tau})\}],
$$

(17)
where
\[ I_i(t) = \begin{cases} 
1, & \text{if reader } i \text{ is still in the system at } t, \\
0, & \text{otherwise.} 
\end{cases} \]  

(18)

Since \( E(\hat{\tau}) = E(\tilde{\tau}) \), (17) implies that
\[ G\{(M_2, y); \tilde{\tau}\} \geq G\{(M_1, y); \hat{\tau}\} = G(M_1, y). \]

From this it follows via (10) that
\[ G(M_2, y) \geq G(M_1, y), \]
as required. ♣

**Theorem 8** If \( R_i \sim IHR \), i.e. \( \Psi_i \), \( 1 \leq i \leq m \), is nondecreasing, then \( G(M, y) \) is nondecreasing in \( y \) when \( M \) is fixed.

**Proof:** We establish the result by means of a contradiction. Hence we suppose that for some \( M \) there exists \( \tilde{y} \) such that
\[ G(M, \tilde{y}) > G(M, \tilde{y} + 1). \]  

(19)

Consider now the reader process in initial state \( (M, \tilde{y}) \). If a single unit of processing is allocated to it, then \( M(1) \subseteq M \) and so by Theorem 7,
\[ G\{M(1), Y(1)\} = G\{M(1), \tilde{y} + 1\} \leq G(M, \tilde{y} + 1) < G(M, \tilde{y}) \]
almost surely,  

(20)
i.e., the Gittins index decreases with probability 1. But by a simple development of Proposition 4.2 in Gittins [3], this implies that
\[ G(M, \tilde{y}) = G\{(M, \tilde{y}); \tau_1\} = \sum_{i \in M} c_i \Psi_i(\tilde{y}). \]  

(21)

However, since \( R_i \sim IHR \), \( 1 \leq i \leq m \), it follows from (21) that
\[ G\{(M, \tilde{y} + 1); \tau_1\} = \sum_{i \in M} c_i \Psi_i(\tilde{y} + 1) \geq \sum_{i \in M} c_i \Psi_i(\tilde{y}) = G(M, \tilde{y}). \]  

(22)

Now, from (10) and (22) we infer that
\[ G(M, \tilde{y} + 1) \geq G(M, \tilde{y}), \]  

(23)

which contradicts (19). This concludes the proof. ♣

**Remarks:**

1. Since optimal policies always choose an option whose Gittins index is maximal, the implication of Theorem 8 is that when \( R_i \sim IHR \), \( 1 \leq i \leq m \), an optimal policy can only preempt the parallel processing of the readers at epochs at which readers leave the system.
2. Although the above analysis is performed under the assumption of integer-valued processing time, standard limiting arguments yield equivalent results for continuous distributions. See Gittins ([3], Chapter 5). We illustrate by an example.
Example 2: Suppose the processing time of the \( i^{th} \) reader is exponentially distributed with mean \( r_i \), while that of the \( j^{th} \) writer is exponentially distributed with mean \( w_j \). It follows easily, from the appropriate continuous time version of (8) that the Gittins index for writer \( j \) is \( d_j/w_j \). Similarly, it follows from a continuous time version of Theorem 6 that the Gittins index for the set \( M \) of unfinished readers is \( \sum_{i \in M} c_i/r_i \). Thus the optimal scheduling policy in this case is as follows: Let \( M \) be the set of unfinished readers, and \( N \) be the set of unfinished writers. Define

\[
GR(M) = \sum_{i \in M} c_i/r_i,
\]

and

\[
GW(N) = \max_{j \in N} d_j/w_j.
\]

If \( GR(M) > GW(N) \), work on all the unfinished readers simultaneously until one of them finishes; else work on the writer with largest \( d_j/w_j \) ratio until it completes. Recompute at the next completion time.

Consider the further simple case where \( r_i = r \) and \( c_i = c \) for all \( 1 \leq i \leq m \), and \( w_j = w \) and \( d_j = d \) for all \( 1 \leq j \leq n \). In this case there is a critical number \( m^* = dr/wc \) such that it is optimal to process readers as long as there are at least \( m^* \) readers in the system, otherwise it is optimal to process writers. When there are no writers left, process all the remaining readers in parallel.

5 Conclusions and Extensions

In this paper we have obtained optimal preemptive and nonpreemptive schedules for reader-writer systems with no arrivals. One case in which it is relatively straightforward to extend the results to allow for arrivals is the preemptive model in Section 4. However, to obtain simple index results of the kind discussed here, new arrivals must be writers.

Consider, then, a modification of the model discussed in Section 4 in which there are \( n \) classes of writers. Members of class \( j \) have iid processing times with common distribution \( W_j \), and a constant cost rate \( d_j \). Moreover, writers of class \( j \) arrive in the system according to a Poisson process with rate \( \lambda_j \). All arrival streams are independent of each other and of the processing times. The readers are as in the original model. No new readers arrive over time.

Such a model can be formulated as a branching bandit (see Weiss [15]) and, more particularly, can be viewed as a special case of Klimov’s problem (see Klimov [5]). The main result here is that the optimal schedule is independent of the arrival rates and hence is the same as that for the version of the model without arrivals. In consequence, the index-based solutions of Section 4 provide optimal policies for this more general model.

The above extension notwithstanding, several problems remain open. It will be useful to relax the condition (i) of Section 3 for the nonpreemptive scheduling problem. However, it seems difficult to do so. Similarly, obtaining closed forms and/or structural properties of the Gittins indices for the reader process in the preemptive case is difficult beyond the special cases discussed in Section 4. However, computational approaches are available quite generally. See Glazebrook [4].
Plainly of interest would be the incorporation of a general arrival stream, to include readers as well as writers. We conjecture that Gittins index policies (suitably defined) will continue to perform well for open problems even when they cannot be shown to be strictly optimal. See, for example, Fay and Glazebrook [2]. Any analysis of the nonpreemptive case with arrivals seems difficult.

**Acknowledgements**

The authors wish to thank the anonymous referees for suggestions that improved the presentation of this paper considerably.

**References**


