Control of a Single-Server Tandem Queueing System with Setups

Izak Duenyas
Diwakar Gupta
Tava Lennon Olsen
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109

Technical Report 95-19
Long Version Containing Proofs
September 1996

CONTROL OF A SINGLE-SERVER TANDEM QUEUEING SYSTEM WITH SETUPS

IZAK DUENYAS
Department of Industrial and Operations Engineering
University of Michigan, Ann Arbor, MI, 48109

DIWAKAR GUPTA
School of Business, McMaster University
1280 Main Street West, Hamilton, Ontario L8S 4M4, Canada

TAVA LENNON OLSEN
Department of Industrial and Operations Engineering
University of Michigan, Ann Arbor, MI, 48109

Abstract

This paper considers the control of a single server tandem queueing system with setups. Jobs arrive to the system according to a Poisson process and are produced to order. A single server must perform a number of different operations on each job. There is a setup time for the server to switch between different operations. We assume that there is a holding cost at each operation which is non-decreasing in operation number (i.e., as value is added to a job, it becomes more expensive to hold). The control problem is to decide which job the server should process at each point in time.

We formulate this control problem as a Markov-Decision Process. We partially characterize the optimal policy, develop an exact analysis of exhaustive and gated polling policies and develop an effective heuristic policy. The results of a simulation study, which tests the performance of the policies considered are reported. These computational results indicate that our heuristic is effective for a wide variety of cases.

1 Introduction

Consider a work center where jobs of a single type arrive and are produced to order. A single server must perform a number of different operations on each job. There is a setup (or switchover) time for the server to switch between different operations. We assume that there is a linear holding cost at each operation which is non-decreasing in operation number. In other words, as more value is added to a job, it becomes no less expensive to hold.

This type of environment is present in many different manufacturing companies, particularly small companies where equipment or labor is limited. For example, consider a workshop where the parts being produced require turning, parting and threading, all on the workshop's only lathe. It takes time to switch the lathe from one type of operation to the next. Or, consider a screen printing operation where a single worker must produce garments with multiple colors. A setup is required each time a new color is printed. Katayama [19] also gives examples of such systems in computer and telephone switching systems.

Netto [27] and Nair [26] analyze the performance of an exhaustive polling policy (or "zero-switching rule") for a two-stage tandem queue where switchover times are zero. Under this policy, the server serves a queue until it becomes empty and then it switches to the next queue. Katayama [15] provides an analysis of the same system with non-zero switchover times. Katayama [16], [17], and [18] provide extensions of this work to gated service, K-limited service and a finite intermediate waiting room, respectively. Both gated and K-limited service strategies are described later in this paper. Work on the N-stage tandem queue includes the paper by Murakami and Nakamura [25] who analyze a 3-stage tandem queue and Konig and Schmit [22] who investigate the relationships between time-stationary and customer-stationary queue length characteristics for these systems. Katayama [19] derives expressions for mean sojourn times and mean waiting times for an N-stage tandem queue with zero switchover times under exhaustive, gated and K-limited service.

There has been very little work on the optimal control of tandem queues. Katayama [20] considers a two-stage multi-class tandem queue with feedback. The server serves the two-stages in an exhaustive and cyclic manner. The control problem is to decide in which order the different classes of customers should be served at the second stage. In a recent parallel work, Iravani et al. [13] consider a two-stage tandem queue and analyze the performance of several different policies including 1) that of serving a predetermined number of jobs in stage 1, and serving stage 2 exhaustively, and 2) a class of dynamic policies with double threshold for switching from stage 1 to 2. Sidi et al. [30] consider a tandem network with a cyclic server similar to ours. In their model, jobs can arrive to any queue from the outside and

the jobs join other queues or leave the system probabilistically after being served at any queue. However, they assume that when the system is empty, the server still cycles the system, setting up each queue one after another. In contrast, we assume that when the system is empty, the server idles at queue 1. Since in our problem, outside arrivals can only occur to queue 1, there is no point in setting up queues $2, \ldots, N$, when no jobs have been processed at queue 1. At such moments, these queues (i.e., queues $2, \ldots, N$) will be empty as well.

There has been some recent work on the optimal control of "polling systems" in which a single server serves multiple queues (see Takagi [33] for an extensive review of polling system literature). In this case, jobs from N different classes arrive to N different queues and require service from the single server. However, unlike in the tandem-queue problem that we are considering, each job requires a single service operation from the server and leaves the system once it is served (rather than join another queue as in the present paper.) The scheduling and control of such systems has been addressed in Boxma [2], Boxma et al. [3], Browne and Yechiali [4], Duenyas and Van Oyen [6], [7], Gupta et al. [11], Hofri and Ross [12], Rajan and Aggrawal [28], Reiman and Wein [29], and Liu et al. [24].

In this paper, we consider the control of a single-server tandem queueing system with N queues and setup times required to switch between each queue. Section 2 presents the problem formulation and introduces notation. In Section 3, we formulate the optimal control problem as a Markov Decision Process (MDP) and partially characterize the optimal policy. Because the dynamic programming approach used for computing the optimal policy suffers from the "curse of dimensionality" when the number of operations is greater than three, heuristics are required for systems with more operations. One possible heuristic policy is to use the "exhaustive polling policy" which serves each queue until it is exhausted, and then switches to the next queue. Another possibility is a gated policy where the server only serves the jobs that it finds at queue 1 upon arrival to queue 1 and then switches to queue 2 and serves all jobs at queues 2, 3, ..., N and switches back to queue 1 again. Section 4 describes these policies further and presents a numerical algorithm for calculating the throughput time, waiting times at each station, and the average cost under these policies. In Section 5, we present a simple heuristic for the control of this system. Section 6 is devoted to a simulation study which compares the performance of our heuristic and the exhaustive and gated polling policies against the optimal policy for problems with 2 and 3 stages and against each other for larger problems. Section 7 concludes the paper.

2 Problem Formulation and Notation

We consider a system with N stations with each station representing an operation to be performed on the job. Customers (or jobs) arrive at station 1 according to an independent Poisson process with rate λ. After receiving service at station i, $i < N$, each customer undergoes a metamorphosis and changes class to $i + 1$. It then moves downstream to station $i + 1$ without incurring any delay. Customers at station N exit the system after receiving service. Thus, class N is the final stage in the processing of the customer. Each job waiting or being processed at station i incurs costs with rate h_i, and we assume that h_i is non-decreasing in i.

Station buffers are assumed to be infinitely large. The service time for a customer at station i is an independent random variable B_i with mean $b_i = E[B_i]$ and rate $\beta_i = 1/b_i$. The time taken by the server to set up for service at station i, is an independent random variable, S_i with mean $s_i = E[S_i]$ and rate $\psi_i = 1/s_i$. We assume that neither setups nor services can be preempted. The traffic intensity at station i is denoted by $\rho_i = \lambda b_i$, and $\rho = \sum_{i=1}^{N} \rho_i$ denotes the server utilization. We assume that $\rho < 1$ since when this holds, there exist policies that result in finite average cost per unit time. This follows from, for example, Altman et al. [1], who show that $\rho < 1$ is both a necessary and a sufficient condition for existence of the stationary joint distribution of queue lengths in polling models with exhaustive and gated service policies.

A policy specifies, at each decision epoch, that the server either remains working in the present queue, idles in the present queue, or sets up another queue for service. The set of decision epochs is assumed to be the set of all arrival epochs, service completion epochs, and setup completion epochs. With $\mathbb{R}^+(\mathbb{Z}^+)$ denoting the nonnegative reals (integers), let $\{X_i^g(t) : t \in \mathbb{R}^+\}$ be the right-continuous queue length process of queue i under policy g (including any customer of queue i in service). Denote the vector of initial queue lengths by $X(0^-) \in (\mathbb{Z}^+)^N$, where $X(0^-)$ is fixed. Without loss of generality, we assume that node one has been set up prior to time $t = 0$ and that the server is initially placed in node one. The average cost per unit time of policy g, $\bar{J}(g)$, can now be expressed as

$$\bar{J}(g) = \limsup_{T \to \infty} \frac{1}{T} E\left\{ \int_0^T \sum_{i=1}^{N} h_i X_i^g(t) dt \right\}. \tag{2.1}$$

The total expected discounted cost of policy g with discount parameter α is expressed as

$$J_\alpha(g) = E\left\{ \int_0^\infty \left(\sum_{i=1}^{N} h_i X_i^g(t) \right) e^{-\alpha t} dt \right\}. \tag{2.2}$$

The objective of the discounted cost optimization problem is to find a policy that minimizes $J_\alpha(g)$, while the objective of the average cost problem is to find a policy that minimizes $\bar{J}(g)$.

3 On an Optimal Policy

In this section, we provide a partial characterization of an optimal policy for the control of a single server tandem queue. In general, the structure of the optimal policy is extremely complicated. For example, even in a problem with as few as three queues (see example at the end of this section), the optimal policy at queue 1 is defined by very complicated (and not necessarily monotonic) switching curves. In general, it is difficult to implement such a complicated policy in a practical setting. However, we provide a partial characterization of the structure to serve as a guide in our efforts to develop an effective yet simple heuristic. We first define the following:

Definition 1: A policy serves node i in a *greedy* manner if the server never idles in queue i while jobs are still available in i and queue i has been set up for service.

Definition 2: A policy serves node i in an *exhaustive* manner if it never switches out of node i while jobs are still available in i.

Definition 3: A *top-priority* queue refers to any queue (there may be more than one) that is served in a greedy and exhaustive manner.

We can then state the following result.

Theorem 1 *For both the discounted and average cost versions of the single-server tandem queue problem, there exists an optimal policy where queue N (the last queue) is a top priority queue* [1].

Proof: Suppose that g is a policy which does not exhaust queue N when there is at least one job at queue N and the server is set up for queue N, and instead chooses to switch to another queue or idle. We let the time when this action is taken by policy g be $t = 0$. Let $t(l)$ denote the time at which the l^{th} control action is taken under policy g. Finally, let L denote the stage, or index of the decision epoch, at which the server serves queue N for the first time. Thus, the server will start to serve queue N for the first time at time $t(L)$. Let the duration of the first job's service at queue N equal B_N. Now, along each sample path of the system, we construct a policy g' which at time $t = 0$ serves the job in node N that is served under policy g at $t(L)$, which possesses the processing time B_N. During $[B_N, B_N + t(L)]$,

[1] This result is also shown in Iravani et al. [13] in parallel work to ours.

g' mimics the actions taken by g during the first $L-1$ stages $[0, t(L))$. At time $t(L+1) = t(L) + B_N$, both g and g' reach the same state along any realization, and g' mimics g from that point on. In order to compute the difference in expected discounted reward of policy g' with respect to g, we define C_l as the change in cost rate that the completion of action l by the server incurs. We note that any action that causes a job to move from station i to $i+1$ causes a cost change of $h_{i+1} - h_i$. A completion of a job at station N causes a cost change of $-h_N$. Finally, a completion of a period in which the server idled or set up causes no change in costs. In order to compare the costs of the policies, we are focusing only on the cost changes caused by the actions of the policies and not by any arrivals to the system. We can then write the cost difference between policy g and g' as

$$J_\alpha(g) - J_\alpha(g') = h_N \int_{B_N}^{B_N + t(L)} e^{-\alpha t} dt + \sum_{l=1}^{L-1} C_l \int_{t(l)}^{t(l)+B_N} e^{-\alpha t} dt. \tag{3.3}$$

The first term on the right-hand-side of (3.3) is due to the fact that policy g' finishes processing the job at node N at (random) time B_N while g finishes processing it at time $B_N + t(L)$. The second term is due to the fact that policy g' completes processing at all other nodes B_N time units later than g. Because $C_l \geq 0$ for $l = 1, \ldots, L-1$ (as the only actions that g completes at these stages are actions that increase costs or keep them constant), and $h_N \geq 0$, we can conclude that policy g incurs a cost that is greater than or equal to policy g' and cannot be optimal. We note that the argument is not dependent on policy g *ever* serving queue N. If policy g never serves queue N, then the upper limit of the first integral in (3.3) is replaced by infinity, and the summation in the second term is an infinite sum. However, as all the terms are still positive, the same argument applies. The proof for the average cost case is similar and is omitted. □

The optimal policy can be fully characterized when all of the setup times are zero, and the system is initially empty. Johri and Katehakis (1988) have shown that when the setup times are zero, the policy where each job is processed through stations 1 through N consecutively (i.e., the server processes one job at each station) stochastically minimizes the total number of customers in the system. It is also clear that processing jobs in this manner minimizes the average number of customers in stations $2, 3, \ldots, N$. Since $h_1 < h_2 < \ldots < h_n$, it therefore also follows that this policy also minimizes the average cost per unit time. (On the other hand, in the discounted case, it might be optimal to idle forever if the number of stages (or the processing times at some stages) and the discount factor are large enough. To see that this is the case, consider a very simple two-stage problem with zero setup times, no arrivals and one job at queue 1 at time 0. Let $h_1 = 1$, and the discount factor equal 0.8. The processing time at stage 1 is

zero, and at the second stage it is one. In this case, when h_2 is greater than 1.83, it is actually better to idle at station 1 than to try to process the job at queue 1.) When $h_1 \geq h_2 \geq \ldots h_N$, and $\beta_i = \beta$ for all i, it is similarly straightforward to show that the optimal policy is to serve the (nonempty) queue i with the highest index $h_i - h_{i+1}$ (where h_{N+1} is defined to be equal to zero). However, in all the applications that we consider, the holding costs are increasing rather than decreasing; we therefore focus on that case for the rest of the paper.

Theorem 1 states that when the server is at the last queue, the number of jobs in other queues do not matter. The optimal policy is to serve the last queue until it is exhausted. Unfortunately, when the server is at any of the other queues, the optimal policy depends on the number of jobs at all queues. To demonstrate this dependence, even in the simplest possible case, we consider a 2-station model with general processing, service and setup times, and the average cost criterion. Under this criterion, server idling is never optimal unless the system happens to be empty and the server happens to be at station 1. (The fact that the server will never idle at queue 2 when queue 2 is not empty follows from Theorem 1; once queue 2 is exhausted, it is optimal to switch to queue 1 immediately since all arrivals occur to queue 1). Because an optimal policy will serve queue 2 exhaustively, we are interested in the optimal control decisions when the server is set up for queue 1. Our purpose is to investigate the dependence of decisions at queue 1 on the number at both queue 1 and queue 2 with this simple model and to use the results in developing heuristics.

Let $V(X_1, X_2, i)$ denote the relative value function of having X_1 jobs in queue 1, X_2 jobs in queue 2, and the server set up for queue $i, i = 1, 2$. We arbitrarily let $V(0, 0, 1) = 0$ and let g denote the average cost rate (gain). We also let $A_1(t)$ denote the (random) number of arrivals to queue 1 during a time duration of t units. The relative value function for this Semi-Markov Decision Process satisfies the dynamic programming optimality equations:

$$
\begin{aligned}
V(X_1, X_2, 1) &= \min \left\{ \begin{array}{l} E[\int_0^{B_1} [h_1(X_1 + A_1(t)) + h_2 X_2] dt + V(X_1 + A_1(B_1) - 1, X_2 + 1, 1)] - g/\beta_1 \\ E[\int_0^{S_2} [h_1(X_1 + A_1(t)) + h_2 X_2] dt + V(X_1 + A_1(S_2), X_2, 2)] - g/\psi_2 \end{array} \right\} \text{for } X_1 \geq 1 \\
V(X_1, X_2, 2) &= E[\int_0^{B_2} [h_1(X_1 + A_1(t)) + h_2 X_2] dt + V(X_1 + A_1(B_2), X_2 - 1, 2)] - g/\beta_2 \quad \text{for } X_2 \geq 1. \\
V(0, X_2, 1) &= \min \left\{ \begin{array}{l} (h_2 X_2)/\lambda + V(1, X_2, 1)] - g/\lambda \\ E[\int_0^{S_2} [h_1(A_1(t)) + h_2 X_2] dt + V(A_1(S_2), X_2, 2)] - g/\psi_2 \end{array} \right\} \\
V(X_1, 0, 2) &= E[\int_0^{S_1} [h_1(X_1 + A_1(t)) dt + V(X_1 + A_1(S_1), 0, 1)] - g/\psi_1
\end{aligned}
\tag{3.4}
$$

We are now ready to state the structure of the optimal policy in this case.

Theorem 2 *For the two-station problem defined by (3.4), the optimal policy when the server is set up for station 1 is completely described by a switching curve $f(X_1)$ such that if $X_2 > f(X_1)$, the optimal policy is to set up for queue 2, otherwise it is to serve queue 1 (or idle when $X_1 = 0$). Furthermore, $f(X_1)$ is non-decreasing in X_1.*

Proof: The proof is given in the Appendix. □.

We note that for systems with more than two stations, the optimal policy has a very complicated structure. For example, in a 3-station problem, the decision at station 2 depends on the number of customers in stations 1 and 3. To illustrate this we present the optimal solution (found using dynamic programming) for the following three station model. All stations have exponential service times with mean one and exponential setup times also of mean one. The arrival rate to queue one is 0.2666667 (resulting in a load (excluding setups) of 0.8). The holding costs are 10 at station 1, 20 at station 2, and 30 at station 3. The structure of the optimal policy is too complicated to display in general so instead we present a specific example that illustrates the dependence of the decision at 2 on both the number of jobs at station 1 and the number of jobs at station 3.

1. When the server is set up for station 2 and station 1 has 3 jobs and station 3 has 10 jobs, the server will switch to 3 if station 2 is empty, stay at 2 if station 2 has 1 to 4 jobs, switch to 3 if station 2 has 5 to 9 jobs, and stay at 2 if station 2 has 10 or more jobs.

2. When the server is set up for station 2 and station 1 has 8 jobs and station 3 has 10 jobs, the server will switch to 3 if station 2 is empty, stay at 2 if station 2 has 1 to 7 jobs, switch to 3 if station 2 has 8 to 9 jobs, and stay at 2 if station 2 has 10 or more jobs.

3. When the server is set up for station 2 and station 1 has 3 jobs and station 3 has 9 jobs, the server will switch to 3 if station 2 is empty, stay at 2 if station 2 has 1 to 5 jobs, switch to 3 if station 2 has 6 to 8 jobs, and stay at 2 if station 2 has 9 or more jobs.

This complicated structure may, at first, appear counter-intuitive but it can be explained. If station 2 has only a few jobs then it is worthwhile for the server to complete these before moving on. However if there are more jobs, then the server will be more likely to leave some jobs behind and move on to the expensive jobs at 3. But, if there are too many jobs at 2 then the cost of making these jobs wait while the server

incurs unnecessary switchover time is too high and the server will remain at 2. However, determining (other than numerically) how many jobs is "too many" or "a few" seems impossible.

Not only must the optimal policy be found numerically, but also, in general, applying such complicated policies is very difficult. This difficulty leads us to consider simpler policies. First, in the next section, we analyze the performance of exhaustive and gated polling policies. These policies are interesting because (a) they are simple to implement, and (b) we can obtain exact results for their performance and hence they can serve as straw policies when we compare them to more complicated policies. Then, in Section 5, we develop an effective heuristic policy.

4 Analysis of an "Exhaustive/Gated Polling" Policy

This section analyzes the performance of an "exhaustive/gated polling" policy for our single server tandem queueing system. Under this policy, stations are visited in the fixed cyclic sequence $1, 2, \cdots, N, 1, \cdots$. Customers present at a station, if any, are served in an *exhaustive* or *gated* fashion immediately after the server finishes a set up for that station.

This section is organized as follows. Section 4.1 describes the policy in more detail and introduces the notation needed. Section 4.2 sets up some preliminary expressions needed to analyze the system. Section 4.3 demonstrates how to find the number of customers present at a polling instant using the computationally efficient descendant sets method. These results are used in section 4.4 to obtain explicit expressions for station 1 mean waiting times. Lastly, Section 4.5 shows how to find mean waiting times at stations 2, 3, $\cdots$, N as a function of the station 1 mean waiting time. The expected cost per unit time under the exhaustive or gated policies can be computed using these mean waiting times and Little's law. In the interest of brevity, we provide detailed arguments only for the exhaustive policy and simply report the equivalent mathematical expressions for the gated policy.

4.1 The Model and Notation

When the server arrives at a station it registers a *server-arrival epoch.* The moment service begins at a station is called a *polling epoch.* The polling and server arrival epochs coincide whenever the polled station has at least one waiting customer. Thus, owing to the downstream movement of type 1 customers, polling and server arrival epochs always coincide at stations 2, 3, $\cdots$, N. In other words, stations 2, 3, $\cdots$, N are never empty at a server-arrival instant at any of these stations. In contrast, upon arriving at

station 1 the server may find no waiting customers and in that case it will idle there until the next arrival occurs.

The possibility of server idling makes our model fundamentally different from related polling models that consider customer routing, for example Sidi et al. [30]. Although the model we consider here is a special case of customer routing in which customers always move downstream, our approach can be modified to accommodate any general probabilistic routing. Analysis of our model is more difficult when compared to non-idling server models because we need to find the idling probabilities and modify functional equations to accurately capture server idling behavior. Furthermore, while the non-idling server model can be obtained as a special case of ours by setting the idling probabilities to zero, the opposite is not true, i.e., the model with server idling cannot be obtained from the non-idling server model. The relationship between these two types of models, in absence of customer routing, has been discussed in detail in two recent papers: Eisenberg [8] and Srinivasan and Gupta [32].

Following the polling instant, the server either serves exactly the number of customers present at the polling instant (gated service) or continues to serve customers until the station queue is empty (exhaustive service). Notice that there is no difference between these two regimes at stations 2, 3, $\cdots$, N which receive only as many customers as are served at station 1. Put differently, only station 1 is affected by the choice of service strategy.

When the server empties a station's queue, it immediately registers a *server-departure epoch.* The server-departure epoch is followed, in turn, by the server-arrival epoch to the next station, after the passage of the appropriate setup time.

It is implicitly assumed that an index used for summation over the stations is (a) replaced by 1 if the index becomes $N+1$ and (b) replaced by N if it drops down to 0. We adopt the convention that an empty product equals 1, and that an empty sum equals 0. For any random variable A, the first two moments are denoted by $E[A]$ and $E[A^2]$. The Laplace Stieltjes Transform (LST) of a random variable A is denoted by $A^*(\cdot)$. Any $1 \times N$ vector of elements z_i is denoted by $\bar{z}$.

The waiting time for a customer, which is the time it spends at station i before its service begins at that station, is denoted by W_i. For this part of our mathematical analysis, we define $T_i = S_{i+1}$ as the setup time at station $i+1$. This results in more compact expressions. Later when reporting the mean waiting time we shall substitute S_{i+1} in place of T_i as appropriate. By default, we treat the case when the service is provided exhaustively. Whenever needed the corresponding notation for the gated service strategy is shown with a hat. For example, $\hat{W}_i$ denotes station i waiting time under the gated regime.

4.2 Preliminaries

Let n_j denote the queue length at station j and ${}^i(n_1, n_2, \cdots, n_N)$ denote the state of the system (queue lengths) at a station i observation epoch. As noted in section 4.1, we consider three observation epochs: server arrival, polling, and server departure epochs. Consider the *joint* probability that a polling epoch happens to be at station i and that there are n_j customers at station j. Let $f_i(\bar{z})$ denote the probability generating function (PGF) of this joint probability distribution. Similarly, let $g_i(\bar{z})$ and $h_i(\bar{z})$ denote the PGF's of corresponding joint probabilities at a server-departure and a server-arrival epoch respectively. It is clear that the terms $f_i(\bar{1})$, $g_i(\bar{1})$ and $h_i(\bar{1})$ represent, respectively, the probabilities that an arbitrary polling, server-departure and server-arrival epoch happens at station i.

Let $F(z) = f_1(z, 1, 1, \cdots, 1)/f_1(\bar{1})$ denote the PGF of the number of customers found at station 1 when (i.e., *given that*) the server polls that station. Define a time interval during which the server is not busy serving customers at station 1 as a "vacation" from station 1. Note that this vacation includes the setup time and any server idle time during which the server is physically at station 1, but it is not serving any customers. Thus the PGF of the number of customers waiting at station 1 at the end of a randomly selected vacation is $F(z)$. If the service strategy is exhaustive, the Fuhrmann-Cooper [9] decomposition for $M/G/1$ queues with vacations permits us to write (see, for example, Cooper [5, equation 18]):

$$W_1^*(s) = \left[\frac{1 - F(1 - s/\lambda)}{(s/\lambda)F'(1)}\right] \frac{s(1-\rho_1)}{s - \lambda + \lambda B_1^*(s)}, \tag{4.5}$$

where the term outside the square brackets is the well-known Pollaczek-Khintchine transform of the waiting-time distribution in the standard $M/\bar{G}/1$ queue. Also, the term $F'(1)$ is the average queue length at station 1 at the instant it is polled.

Similarly, if the service strategy is gated, the Fuhrmann-Cooper decomposition leads to the following expression (see [31, equation 37] for details):

$$\hat{W}_1^*(s) = \left[\frac{F(B_1^*(s)) - F(1 - s/\lambda)}{(s/\lambda)(1-\rho_1)F'(1)}\right] \frac{s(1-\rho_1)}{s - \lambda + \lambda B_1^*(s)}. \tag{4.6}$$

4.3 The Number of Customers at Polling Instants

¿From (4.5) and (4.6) we observe that the LST of the waiting time at station 1 can be determined if we can evaluate $F(z)$. We shall show in Section 4.5 that the mean waiting time at other stations can be found if we know $E[W_1]$. In this section, we focus on the waiting time at station 1 and thus address the problem of determining $F(z)$. This is the PGF of the number of customers present at station 1 when the server polls that station. Since $F(z) = f_1(z, 1, \cdots, 1)/f_1(\bar{1})$ we actually focus on evaluating $f_1(z, 1, \cdots, 1)$

instead. Our approach is based on the concept of "descendant sets" and leads to a computationally efficient algorithm. This technique has been used in several recent studies to analyze variants of polling systems: Konheim, Levy and Srinivasan [21] analyze systems with a continuously roving server, Srinivasan and Gupta [32] treat the case of a patient server, and Gupta and Srinivasan [10] consider a continuously roving server with state-dependent setups.

4.3.1 The Descendant Sets Approach

Consider a randomly selected point in time at which the server polls station 1, and call this the reference point. Let X_1 denote the number of customers present at station 1 at the reference point. Note that $F(z)$ is the PGF of X_1. Define a cycle as the elapsed time between two successive polling epochs at station 1. Suppose $C_{i,c}$ is one of the customers present at polling station i, c cycles prior to the reference point, where $c = 0, 1, \cdots$. The reference point thus marks the beginning of a cycle with cycle index $c = -1$.

We define the immediate "offspring" of a $C_{i,c}$ as the set of all customers arriving to the system during the service of $C_{i,c}$, and the "descendant set" of $C_{i,c}$ is recursively defined to consist of $C_{i,c}$, its offspring (if any), and any descendants of its offspring. Let the random variable $L_{i,c}$ denote the number of customers present at station 1 at the reference point that are also in the descendant set of $C_{i,c}$, and let $L_{i,c}(z)$ denote its PGF. We will say that customer $C_{i,c}$ "contributes" an amount distributed as $L_{i,c}$ to X_1. Our procedure is based upon the fact that we can express $L_{1,c}$, recursively, in terms of contributions to X_1 made by customers who arrive during each $C_{1,c}$'s service period.

Consider, for example, the exhaustive service strategy. All type 1 customers who arrive during the service period of a $C_{1,c}$ customer are served in the same cycle and so the PGF of their contribution is $L_{1,c}(z)$. Each type 1 customer also contributes by giving birth to exactly one type 2 customer, who is also served in the same cycle and contributes, in turn, $L_{2,c}$ to X_1. Putting it all together, we have,

$$L_{1,c}(z) = B_1^*\left(\lambda - \lambda L_{1,c}(z)\right) L_{2,c}(z), \quad c \geq 0. \tag{4.7}$$

Next, a type j customer generates contributions in two ways. There are external arrivals to station 1 that occur during its service period and are served in the next cycle. Then there are additional contributions that come from the type $(j+1)$ customer which it spawns. This latter arrival is served in the same cycle, i.e., the cycle indexed c. From these arguments we get,

$$L_{j,c}(z) = B_j^*(\lambda - \lambda L_{1,c-1}(z))L_{j+1,c}(z), \quad \text{for all } j = 2, 3, \cdots, N-1. \tag{4.8}$$

At stage N, there is no further metamorphosis of the customer and therefore the above relationship is modified as follows:

$$L_{N,c}(z) = B_N^*(\lambda - \lambda L_{1,c-1}(z)). \tag{4.9}$$

The relations described in equations (4.7), (4.8), and (4.9) allow us to determine the PGF's $L_{i,c}(z)$ recursively, by working backwards from the reference point. Because customers who arrive after the reference point do not contribute to X_1, we have $L_{j,-1} = 0$ for $j > 1$ and, furthermore, $L_{j,c} = 0$ for $c < -1$ and all j. This implies that $L_{j,-1}(z) = 1$ for $j > 1$, and $L_{j,c}(z) = 1$ for all j, if $c < -1$. On the other hand, because each customer present at station 1 at the reference point contributes exactly 1 to X_1, we have $L_{1,-1} = 1$, and $L_{1,-1}(z) = z$. Thus, the recursion begins with the initial condition $(L_{1,-1}(z), \cdots, L_{N,-1}(z)) = (z, 1, \cdots, 1)$.

Now let the random variable $T_{i,c}$ denote the total contribution to X_1 made by *all* those customers who arrive during a setup performed at station $i+1$, c cycles prior to the reference point, and let $T_{i,c}(z)$ denote its PGF. Because the reference point occurs after server-arrival at station 1, $T_{i,c}$ does not contribute to X_1 so long as $c \leq -1$. Hence, $T_{i,c}(z) = 1$, for all i, if $c \leq -1$.

Arguing along lines similar to those that have been used to obtain equations (4.7), (4.8), and (4.9), we can write

$$T_{i,c}(z) = T_i^*\left(\lambda - \lambda L_{1,c-1}(z)\right), \quad \forall\ 1 \leq i \leq N, \quad \text{and } c \geq 0. \tag{4.10}$$

We are now ready to evaluate $f_1(z, 1, \cdots, 1)$. But before doing that, we present equivalent recursive relationships for the gated service regime.

$$\hat{L}_{j,c}(z) = B_j^*(\lambda - \lambda \hat{L}_{1,c-1}(z))\hat{L}_{j+1,c}(z), \quad \text{for all } j = 1, 2, \cdots, N-1. \tag{4.11}$$

$$\hat{L}_{N,c}(z) = B_N^*(\lambda - \lambda \hat{L}_{1,c-1}(z)). \tag{4.12}$$

$$\hat{T}_{j,c}(z) = T_j^*\left(\lambda - \lambda \hat{L}_{1,c-1}(z)\right), \quad \forall\ 1 \leq j \leq N, \quad \text{and } c \geq 0. \tag{4.13}$$

The boundary values for the $\hat{L}_{i,c}$'s and $\hat{T}_{i,c}$'s are the same as those for the $L_{i,c}$'s and $T_{i,c}$'s.

4.3.2 Evaluating $f_1(z, 1, \cdots, 1)$

To see how the above discussion relates to evaluating $f_1(z, 1, \cdots, 1)$, we first note this term is the same as $f_1(L_{1,-1}(z), L_{2,-1}(z), \cdots, L_{N,-1}(z))$. Next we note that $f_1(L_{1,c}(z),\ L_{2,c}(z),\ \cdots,\ L_{N,c}(z))$ denotes the PGF of the sum of the contributions to X_1 made by all customers present in the system when the server

polls station 1, c cycles prior to the reference point. It will be notationally convenient to denote this PGF by $f_{1,c}(z)$, i.e., we define

$$f_{1,c}(z) \stackrel{\Delta}{=} f_1(L_{1,c}(z), L_{2,c}(z), \cdots, L_{N,c}(z)). \tag{4.14}$$

Similarly, we define $g_{i,c}(z)$ ($h_{i,c}(z)$) as the PGF of the sum of contributions made to X_1 by the customers present at the various stations at the server-departure epoch (server-arrival epoch) at station i, c cycles prior to the reference point. Thus, we have

$$g_{i,c}(z) \stackrel{\Delta}{=} g_i(L_{1,c-1}(z), 1, \cdots, 1, L_{i+1,c}(z), 1, \cdots, 1), \quad \text{for all } i, \text{ and,} \tag{4.15}$$

$$h_{i,c}(z) \stackrel{\Delta}{=} h_i(L_{1,c-1}(z), 1, \cdots, 1, L_{i,c}(z), 1, \cdots, 1) \quad \text{for } j \geq 2. \tag{4.16}$$

The "1" in the argument on the right hand side of above relationships signifies empty queues, which contribute nothing and therefore the GF of contributions is 1. Finally, for station 1 we define

$$h_{1,c}(z) \stackrel{\Delta}{=} h_1(L_{1,c}(z), 1, \cdots, 1). \tag{4.17}$$

Now consider the customers present at the various stations at a server departure epoch from station i, c cycles prior to the reference point. At such an observation epoch, only stations 1 and $i+1$ may have some customers present, all other station queues must be empty. Those at station 1 will each contribute to X_1 an amount with PGF $L_{1,c-1}(z)$ while those at station $i+1$ will each contribute an amount with PGF $L_{i+1,c}(z)$. It is clear that a server arrival to station i always follows the server-departure epoch from station $i-1$, for all $i = 1, 2, \cdots, N$. Thus, additional contributions to X_1 come from arrivals during the setup time T_i, and

$$\begin{aligned} h_{i,c}(z) &= g_{i-1,c}(z)T_{i-1,c}(z) \quad \text{for all } i = 2, 3 \cdots, N, \quad \text{and} \\ h_{1,c}(z) &= g_{N,c}(z)T_{N,c}(z). \end{aligned} \tag{4.18}$$

At stations 2, 3, $\cdots$, N, the contribution to X_1 at a server-departure epoch remains exactly what it was at the server-arrival epoch to that station, because the contribution from offspring of those which are present at various queues at the server arrival instants is already accounted for. This implies,

$$g_{i,c}(z) = h_{i,c}(z) \quad \text{for all } i = 2, 3 \cdots, N. \tag{4.19}$$

In contrast, at station 1, the polling instant starts a new cycle and occurs either immediately after the server arrival to station 1 (this happens whenever station 1 queue is not empty) or as a result of an

external arrival following a period of server idleness. If an external arrival causes the polling instant to occur, then this arrival contributes $L_{1,c-1}$ to X_1, because it is served in the next cycle. Hence,

$$f_{1,c-1}(z) = h_{1,c}(z) - h_1(\bar{0}) + h_1(\bar{0})L_{1,c-1}(z), \tag{4.20}$$

where, by stationarity, the probability that the system is empty at a server-arrival epoch at station 1 is independent of the cycle being observed and we have written $h_{1,c}(0) = h_1(\bar{0})$. Finally, because $g_{1,c}(z) = f_{1,c}(z)$ we get

$$g_{1,c}(z) = h_{1,c+1}(z) - h_1(\bar{0}) + h_1(\bar{0})L_{1,c}(z). \tag{4.21}$$

Because $h_{i,c}(z) \mid_{z=1} = h_i(\bar{1})$, by putting $z = 1$ in equations (4.18), (4.19), and (4.21) and noting that $T_{i,c}(1) = 1$ we see that $h_i(\bar{1}) = h_{i-1}(\bar{1})$. Similarly, from equations (4.19) and (4.21), we also see that $h_i(\bar{1}) = g_i(\bar{1})$ and that $g_i(\bar{1})$ is independent of i. Intuitively, this makes sense because there is exactly one server-departure and server-arrival at each station during each cycle. Next, if we consider that $\sum_{i=1}^{N} g_i(\bar{1}) = 1$, it follows immediately that $g_i(\bar{1}) = 1/N$ for any i and

$$f_1(\bar{1}) = g_1(\bar{1}) = 1/N. \tag{4.22}$$

¿From (4.18) and (4.19) we see that for $i \geq 2$, $g_{i,c}(z) = g_{i-1,c}(z)T_{i-1,c}(z)$. Using this observation and equations (4.18) and (4.21) we can write,

$$g_{1,c}(z) = g_{1,c+1}(z) \prod_{j=1}^{N} T_{j,c+1}(z) - h_1(\bar{0})J_{N,c}(z), \tag{4.23}$$

where we define $J_{N,c}(z) = 1 - L_{1,c}(z)$. Starting with $g_{1,-1}(z)$, writing it in terms of $g_{1,0}(z)$, and continuing in this fashion for m times, we get:

$$f_{1,-1}(z) = g_{1,-1}(z) = g_{1,m-1}(z) \prod_{c=0}^{m-1} \prod_{i=1}^{N} T_{i,c}(z) - h_1(\bar{0}) \sum_{c=-1}^{m-1} J_{N,c}(z) \prod_{p=0}^{c} \prod_{k=1}^{N} T_{k,p}(z). \tag{4.24}$$

In the limit, as $c \to \infty$, it can be shown that $L_{i,c} \to 0$ (i.e., the contribution to X_1 made by a customer who is served c cycles ago tends to 0 as c tends to infinity) and therefore $L_{i,c}(z) \to 1$. This implies that as $c \to \infty$, $g_{1,c}(z) \to g_1(\bar{1}) = 1/N$ (from equation (4.22)). Hence, we have that

$$f_1(z, 1, \cdots, 1) = f_{1,-1}(z) = (1/N) \left[\prod_{c=0}^{\infty} \prod_{i=1}^{N} T_{i,c}(z) - \vartheta_1 \sum_{c=-1}^{\infty} J_{N,c}(z) \prod_{p=0}^{c} \prod_{k=1}^{N} T_{k,p}(z) \right], \tag{4.25}$$

where

$$\vartheta_1 = h_1(\bar{0})/g_1(\bar{1}) = N h_1(\bar{0}), \tag{4.26}$$

represents the ratio of the number of empty server-arrival epochs registered by the server at station 1 for every departure it makes from station 1 to station 2.

¿From equation (4.25), it is clear that we can evaluate the PGF of X_1 if the unknown ϑ_1 can be determined. To find ϑ_1, we use equation (4.25) and the fact that $f_1(0,1,\cdots,1) = 0$, i.e., a station 1 polling instant does not occur if queue at station 1 is empty. This yields,

$$\vartheta_1 = \frac{\prod_{c=0}^{\infty}\prod_{i=1}^{N} T_{i,c}(0)}{\sum_{c=-1}^{\infty} J_{N,c}(0)\prod_{p=0}^{c}\prod_{k=1}^{N} T_{k,p}(0)}. \tag{4.27}$$

It is interesting to note that the expression for the PGF of the number of type 1 customers at a station 1 polling instant is exactly the same as given in equation (4.25) even for the gated service strategy.

4.4 Expected Waiting Time at Station 1

Differentiating equations (4.5) and (4.6) with respect to s and setting $s = 0$ yields the following expressions for station 1 mean waiting time under exhaustive and gated regimes, respectively.

$$E[W_1] = \frac{(1-\rho_1)F''(1) + \lambda^2 E[B_1^2]F'(1)}{2\lambda(1-\rho_1)F'(1)}. \tag{4.28}$$

$$E[\hat{W}_1] = \frac{(1+\rho_1)\hat{F}''(1)}{2\lambda\hat{F}'(1)}. \tag{4.29}$$

The terms $F'(1)$ and $F''(1)$ are found by successively differentiating equation (4.25). For the exhaustive service regime, we obtain the following result after considerable simplification,

$$F'(1) = \lambda E[C](1-\rho_1), \tag{4.30}$$

where $E[C] = \{E(\sum_{i=1}^{N} S_i) + \vartheta_1/\lambda\}/(1-\rho)$ denotes the expected cycle length. Similarly, for the gated service regime

$$\hat{F}'(1) = \lambda E[\hat{C}], \tag{4.31}$$

where $E[\hat{C}] = \{E(\sum_{i=1}^{N} S_i) + \hat{\vartheta}_1/\lambda\}/(1-\rho)$. The equations for $F''(1)$ are obtained in a similar fashion and are more complicated. We report below, only the final outcome after substituting from these relationships into the expression for the mean station waiting time and simplifying.

$$\begin{aligned} E[W_1] \;=\; & \frac{(1-\rho_1)(\sum_{i=1}^{N} E(S_i))^2}{2E[C](1-\rho)^2} + \frac{(1-\rho_1)\{\sum_{i=1}^{N}(\lambda E[B_i^2] + Var(S_i)/E[C])\}}{2(1-\rho)(1+\rho-2\rho_1)} \\ & + \frac{\rho_1(\rho-\rho_1)^2 + [\sum_{k=1}^{N-2}\sum_{j=2}^{N-k}\rho_j\rho_{j+k}](1-\rho_1)}{\lambda(1-\rho)(1+\rho-2\rho_1)} + \frac{\vartheta_1(1-\rho_1)(\rho-\rho_1)(\sum_{i=1}^{N} E(S_i))}{\lambda E[C](1-\rho)^2(1+\rho-2\rho_1)}. \end{aligned} \tag{4.32}$$

$$E[\hat{W}_1] = \frac{(1+\rho_1)(\sum_{i=1}^{N} E(S_i))^2}{2E[\hat{C}](1-\rho)^2} + \frac{(1+\rho_1)\{\sum_{i=1}^{N}(\lambda E[B_i^2] + Var(S_i)/E[\hat{C}])\}}{2(1-\rho^2)} + \frac{(1+\rho_1)(\sum_{k=1}^{N-1}\sum_{j=1}^{N-k}\rho_j\rho_{j+k})}{\lambda(1-\rho^2)} + \frac{\hat{\vartheta}_1\rho(1+\rho_1)[\sum_{i=1}^{N} E(S_i)]}{\lambda E[\hat{C}](1-\rho)^2(1+\rho)}. \tag{4.33}$$

Notice that the above expressions are explicit in input parameters except for the ϑ_1 ($\hat{\vartheta}_1$ for gated service) term which needs to be found using the recursive relationship of equation (4.27). This simplification leads to an extremely fast computer algorithm in each case.

4.5 Expected Waiting Times at Stations 2, 3, $\cdots$, N

During any given cycle, the number of customers served at stations 1, 2, $\cdots$, N is exactly the same. Let the random variable Y ($\hat{Y}$ for gated regime) denote this number and the function $Y(z)$ denote the corresponding PGF. Let A denote the size of the batch of customers which is generated by each type 1 customer at a station 1 polling instant under the exhaustive service regime. Notice that all A customers are served in the same cycle. If $A(z)$ denotes the PGF of A, then it follows from arguments similar to those which led to equation (4.7), that

$$A(z) = zB_1^*(\lambda - \lambda A(z)). \tag{4.34}$$

Furthermore, it is now apparent that

$$Y(z) = F(A(z)). \tag{4.35}$$

In contrast, when the service discipline is gated, only as many customers are served in each cycle as are present at the corresponding station 1 polling instant. Thus, in that case

$$\hat{Y}(z) = \hat{F}(z). \tag{4.36}$$

Consider the total accrued waiting time at station j in a cycle in which exactly n customers are served. The waiting time experienced by the $(k+1)^{\text{st}}$ customer is the time it takes to process the $(n-k-1)$ customers behind it at station $j-1$, plus the setup time at station j and the time taken to process the k type j customers in front of it. Upon summing over k we find the conditional accrued waiting time, $TW_{j,n}$, as follows:

$$TW_{j,n} = nS_j + \sum_{k=0}^{n-1}\{(n-k-1)B_{j-1} + kB_j\}, \tag{4.37}$$

First simplifying the above summation and then taking expectations on both sides yields

$$E(TW_j) = E(Y)E(S_j) + \frac{E[Y(Y-1)]}{2}[E(B_{j-1}) + E(B_j)]. \tag{4.38}$$

Next, we substitute for $E(Y)$ and $E[Y(Y-1)]$, and then divide the expected accrued waiting time per cycle with $\lambda E[C]$, the expected number of customers served per cycle, to obtain the following expression for the expected waiting time of type j customers:

$$E(W_j) = E(S_j) + \frac{\lambda E[W_1] + \rho_1}{(1-\rho_1)}[E(B_{j-1}) + E(B_j)]. \tag{4.39}$$

Using similar arguments we also obtain,

$$E(\hat{W}_j) = E(S_j) + \frac{\lambda E[\hat{W}_1]}{(1+\rho_1)}[E(B_{j-1}) + E(B_j)]. \tag{4.40}$$

5 Heuristic Solutions

The structure of the optimal policy is rather complex as the action that the server chooses when it is set up for a given station depends on the number of jobs at other stations in complicated ways. Furthermore, even a problem with only 4 stations will result in a dynamic program with over 50 million states if the number of jobs at a station is truncated at 60. These considerations lead us to develop a simple heuristic in this section.

Under our heuristic policy, the server serves queue 1 until the server has processed K jobs or queue 1 is empty, whichever occurs first. Thus our heuristic policy switches from queue 1 to queue 2 whenever $X_1 = 0$ and $X_2 > 0$, or $X_2 = K$. The server idles at queue 1 when $\sum_{i=1}^{N} X_i = 0$. As was shown in Section 3, the optimal policy for switching from queue 1 to queue 2 involves a more complicated switching curve. However, as we will show in the next section, replacing the optimal switching curve by our much simpler threshold does not affect performance significantly.

In a 2-station problem, stating a threshold K would completely specify the policy because once the server switches to queue 2, using the insight gained from Section 3, we require the server to process all the jobs in queue 2 and then to switch to queue 1 again. However, in a problem with more than 2 stations, we have to specify what the server does next. In general, it might be optimal for the server to switch back to queue 1 before completing all the jobs in queues $2, 3, \ldots, N$. However, we will focus on policies that never switch to queue 1 again until all jobs in queues $2, 3, \ldots, N$ have been served. The server will only return to queue 1 again when $\sum_{j=2}^{N} X_j = 0$. In fact, under our heuristic, the server does not return

to a station until all jobs at stations with a higher index have cleared the system. When holding costs are increasing downstream, as we assume in our model, the optimal policy rarely requires a switch to station 1 before completing jobs at other stations. Furthermore, not permitting the server to return to station 1 until all jobs at other stations have been completed allows us to state a very simple heuristic policy because, once the server is at queue 2, the problem becomes one of deciding how to push the jobs at queue 2 through stations $2, 3, \ldots, N$.

We note that the batch of size K (or less if the server switched to queue 2 before completing K jobs) at queue 2 may be split further at queue 2 and at later stations. That is, the server may decide to split the X_2 jobs that it finds at queue 2 into y_2 batches. Once the first X_2/y_2 jobs have been served, the server will switch to queue 3. The server will not return to queue 2 until the first X_2/y_2 jobs have cleared the system. However, these jobs may be further split in the stations ahead. Therefore, our policy is completely specified by K and a description of the splitting procedure. Our heuristic chooses the "best" splitting procedure for each K and estimates its cost. The value of K with the lowest estimated cost is selected.

We note that if we know how the original batch at queue 2 is split at station 2 and all other stations, we can immediately compute how long the whole batch will take to go through the system. We let y_i equal the number of smaller batches that a batch arriving from the previous station will be split into at station i. If a batch can not be split into equal sized groups, then it is divided as evenly as possible with the extra customers spread out over the earlier batches.

Consecutive batches at a station therefore differ by no more than one customer and our policy is completely described by K and the y_i values. As an example, consider a 5-station problem with $K = 12$, $y_2 = 2, y_3 = 1$, and $y_4 = 2$. The server will switch from queue 1 either when it has processed 12 jobs or when queue 1 is empty. Suppose the server processes 10 jobs at queue 1 and then has to switch to queue 2. At station 2, the server would process a batch of 5, then switch to queue 3, process 5 jobs at queue 3 , switch to queue 4 and process 3 jobs, switch to queue 5, process the 3 jobs, switch back to queue 4 and process 2 jobs, switch to queue 5 and process 2 jobs and then switch back to queue 2 and push the remaining 5 jobs there through stations 2 and 5 in the same way and then switch back to queue 1 again. Table 1 graphically demonstrates the server actions in this case. The first row corresponds to the station number being processed, the second to the number processed at that station, and the last 5 rows to the number of jobs at each station when the current station is switched to.

We decide on the best values of y_i by using a greedy heuristic. We first estimate the cost of processing

Station Number	1	2	3	4	5	4	5	2	3	4	5	4	5
Number Processed	10	5	5	3	3	2	2	5	5	3	3	2	2
# at Station 1	10	0	0	0	0	0	0	0	0	0	0	0	0
# at Station 2	0	10	5	5	5	5	5	5	0	0	0	0	0
# at Station 3	0	0	5	0	0	0	0	0	5	0	0	0	0
# at Station 4	0	0	0	5	2	2	0	0	0	5	2	2	0
# at Station 5	0	0	0	0	3	0	2	0	0	0	3	0	2

Table 1: Example of processing 10 jobs with $y_2 = 2, y_3 = 1$, and $y_4 = 2$.

batches of K jobs with no splits (i.e., $y_i = 1$ for all i). Next, we consider increasing only one of the y_i values (note that there are $N-2$ candidates for an increase of one). We estimate the cost of the policy for each of the $N-2$ candidates. We choose the combination with the minimum cost (using the cost approximation we describe below). We then repeat the process and decide which value of y_i to increase by 1 next. At each iteration, the cost is recorded, and we continue until no more splits are possible. The splitting policy with the minimum estimated cost is chosen as the best policy for the given value of K. (Note that this may be the policy that sends the batch of size K through the whole system.) In this manner, we find the best splitting policy and estimated cost for $K = 1, \ldots, K_{max}$. (K_{max} can be set arbitrarily large depending on how many values the policy maker wants to evaluate.) The minimal cost K value and its associated splitting policy is adopted.

At each iteration of our procedure, we need to compute the expected cost of a policy specified by the value of K and the vector $y = (y_2, y_3, \ldots, y_{N-1})$. The number of jobs at queue 2 when the server switches to queue 2 ranges between 1 and K. We let n_{ij}^K denote the number of batches processed at station i when the server switches to station 2 after processing j jobs at station 1. For example, if $K = 9$ with $y_2 = 2$ and $y_3 = 3$ in a 4-station problem, when a batch of 8 is processed through stations 2, 3, and 4, we will have $n_{2,8}^9 = 2$, $n_{3,8}^9 = n_{4,8}^9 = 6$.

In order to estimate the cost of a policy, we divide the cost into two components. One component is the amount of cost that a job will incur waiting starting from the time it arrives to queue 1 until the server starts processing the batch that this job is a part of at queue 1. The second cost is the amount of cost that a job will incur from the time the server begins processing the batch that this job is a part of until the job has cleared the system. We start by computing the second component.

We let $c_j(K,y)$ denote the expected sum of the holding costs incurred by a batch of size j (from the time the first unit in the batch is processed at queue 1 until the last unit is finished at queue N). This cost can easily be computed exactly using a recursive procedure. The expected cost per job when a batch of size j is processed is then given by $c_j(K,y)/j$. Let $p_j(K,y)$ denote the probability that the server processes a batch of size j at queue 1 before switching to queue 2 when using the policy defined by K and y. Let $w(K,y)$ denote the expected amount of time that a job waits at station 1 before the batch that it belongs to starts being served. Then, our estimate of the average cost per job under policy (K,y) is given by

$$AC(K,y) = h_1 w(K,y) + \sum_{J=1}^{K} p_j(K,y)\frac{c_j(K,y)}{j}. \tag{5.41}$$

Therefore, we need to obtain estimates of $p_j(K,y)$ and $w(K,y)$ in order to compute $AC(K,y)$.

The estimate for $w(K,y)$ is obtained by assuming that batches of size K get processed each time. In fact, we replace the original system with one where jobs arrive only in batches of K. Using our splitting procedure, the amount of time that the server would spend on processing a batch of j through all the stages has mean

$$m_j = \sum_{i=1}^{N}(n_{ij}^K s_i + j b_i), \tag{5.42}$$

and variance

$$v_j = \sum_{i=1}^{N}(n_{ij}^K Var[S_i] + j Var[B_i]). \tag{5.43}$$

Hence, the system where jobs arrive in batches of K and are served in batches of K is a $GI/G/1$ queue with processing time (for a whole batch) with mean m_K and variance v_K. Because in the original system, the interarrival time is assumed to be exponential with rate λ, in this system the interarrival time of a batch of size K is Erlang-K. Let $\rho_K = \lambda m_K/K$, $c_{aK}^2 = 1/K$, and $c_{sK}^2 = v_K/m_K^2$. Then using the waiting time approximation for a $GI/G/1$ queue (see Whitt [35]) our estimate for the waiting time is

$$w(K,y) = \frac{K\rho_K^2(c_{aK}^2 + c_{sK}^2)}{2\lambda(1-\rho_K)}. \tag{5.44}$$

It remains to approximate the $p_j(K,y)$. Our estimate, denoted by $p_j'(K,y)$, is first calculated as the steady-state probability of there being j customers in an $M/M/1$ queue with arrival rate λ and mean service time m_K/K. (Notice that because setup initiations at queue 2 do not occur according to a Poisson process this is only an approximation.) We use probabilities for the $M/M/1$ queue because steady-state probabilities are not easily obtainable for $GI/G/1$ queues. Therefore, the initial approximation for the

probability of processing j jobs is:

$$p_j'(K,y) = \begin{cases} (1-\rho_K)(1+\rho_K) & : \quad j=1 \\ (1-\rho_K)\rho_K^j & : \quad 1<j<K \\ \rho_K^K & : \quad j=K \end{cases} \tag{5.45}$$

Here the probability of processing one job is approximated by the sum of the $M/M/1$ steady-state probabilities of there being zero or one job, and the probability of processing K jobs is approximated by the $M/M/1$ steady-state probability of there being K or more jobs.

However, in the real system, the server often processes fewer than K jobs at queue 1 before switching to queue 2 and hence the expected service time per job will not be m_K/K. It can be easily seen that $m_j/j \geq m_K/K$ for $j < K$ and so the percentage of time the server is busy will be greater than $\rho_K = \lambda m_K/K$. In fact, it will be equal to

$$\rho_K' = \lambda \sum_{j=1}^{K} p_j(K,y) m_j/j. \tag{5.46}$$

Therefore (5.46) above can be further substituted into (5.45) to yield the following system of simultaneous equations:

$$\rho_K' = \lambda \sum_{j=1}^{K} p_j'(K,y) m_j/j.$$

$$p_j'(K,y) = \begin{cases} (1-\rho_K')(1+\rho_K') & : \quad j=1 \\ (1-\rho_K')\rho_K'^j & : \quad 1<j<K \\ \rho_K'^K & : \quad j=K \end{cases}$$

This yields

$$\rho_K' = \lambda \left((1-\rho_K'^2)m_1 + \sum_{j=2}^{K} \rho_K'^j m_j/j - \sum_{j=2}^{K-1} \rho_K'^{j+1} m_j/j \right) \tag{5.47}$$

$$= \lambda m_1 + \lambda \sum_{j=2}^{K} \rho_K'^j \left(\frac{m_j}{j} - \frac{m_{j-1}}{j-1} \right) \tag{5.48}$$

It is clear that (5.48) has a unique solution between 0 and 1 and therefore can be easily solved using standard numerical methods such as bisection. The solution to these equations is then substituted into (5.41) which yields an estimate of the cost of the policy.

Finally, we note that our heuristic is guaranteed not to result in the queue sizes growing to infinity. First, note that under the heuristic policy, the only queue that might have more than K jobs is queue

1 as all other queues have at most K jobs at any time. Whenever there are K or more jobs in queue 1, the server will always serve a batch of jobs at queue 1 and then push these K jobs through the other nodes. Now because $w(K, y)$ needs to be finite (otherwise AC(K,y) would be infinite), the y values are always chosen such that the expected sum of the processing and setup times for K jobs through the whole system is less than K times the expected interarrival time. Therefore, whenever there are more than K jobs, the "effective" service rate is greater than the "effective" arrival rate and queue 1 is guaranteed to eventually have less than K jobs.

6 A Simulation Study

We conducted a simulation study to test the performance of our heuristic. Because systems with more than 3 operations become extremely difficult to solve optimally (as we noted in Section 5, an MDP formulation of a problem with 4 operations can easily require over 60 million states) the heuristic was tested against the optimal policy only for problems with three operations. We also compared our heuristic to cyclic serve-to-exhaustion and gated service for systems with five operations. The simulation was run for 5,000,000 hours and the method of batch-means was used to calculate performance statistics. The confidence intervals for the simulation estimates had a width of about 3% of the mean values.

In Table 2 we present the results comparing the heuristic, exhaustive and gated service to the optimal solution for systems with three stations. The holding costs for all three station examples were 10 at station 1, 20 at station 2, and 30 at station 3. In this case, we considered three cases for the mean service time values. In case 1, the mean service times were completely identical. In case 2, the mean service times were increasing in station index, and in the last case they were decreasing in station index. We considered 7 different mean setup time combinations for each processing time case, resulting in 21 different cases. The heuristic performed very well with an average percentage suboptimality of 2.4% and a maximum suboptimality of 4.6%. In contrast, the exhaustive polling policy's average suboptimality was 44.9% and the gated had an average percentage suboptimality of 26.1%.

In general, we found that the heuristic produced results that were consistent with our intuition. For example, in cases 2, 9 and 16 when setup times are zero the heuristic gives a maximum batch size of one. This is the optimal method of scheduling these types of system. Also, when setup times dropped from one station to the next the heuristic was more likely to form a split at the low setup station. For example, in case 5 in Table 2 the heuristic yields $K = 5$ and $y_2 = 5$.

Table 3 compares the heuristic to the exhaustive and gated service disciplines for systems with five

Case	ρ	b_1	b_2	b_3	s_1	s_2	s_3	heur. (%sub.)	exhaustive (% sub.)	gated (% sub.)	Opt.
1	0.8	1	1	1	1	1	1	149.93 (2.40)	160.58 (9.68)	156.40 (6.82)	146.41
2	0.8	1	1	1	0	0	0	37.42(0.24)	77.75(108.29)	62.62(67.74)	37.33
3	0.8	1	1	1	0.5	0.5	0.5	100.20 (2.24)	117.43 (19.81)	108.23 (10.43)	98.01
4	0.8	1	1	1	2	2	2	246.18 (4.53)	246.91 (4.83)	252.19 (7.08)	235.52
5	0.8	1	1	1	1.5	0	0	79.07(4.63)	112.86 (49.35)	103.49 (36.95)	75.57
6	0.8	1	1	1	0	1.5	0	103.22 (2.22)	120.86 (19.69)	111.49 (10.41)	100.98
7	0.8	1	1	1	0	0	1.5	107.02 (3.12)	124.86 (20.32)	115.49 (11.28)	103.78
8	0.7	1	2	4	1	1	1	57.83(1.76)	64.18(12.93)	62.51(10.00)	56.83
9	0.7	1	2	4	0	0	0	28.90(0.79)	43.87(53.01)	41.02(43.08)	28.67
10	0.7	1	2	4	0.5	0.5	0.5	44.43(2.34)	53.57(23.37)	51.32(18.20)	43.42
11	0.7	1	2	4	2	2	2	85.21(3.98)	86.78(5.89)	86.14(5.11)	81.95
12	0.7	1	2	4	1.5	0	0	38.71(2.31)	51.36(35.74)	49.12(29.81)	37.84
13	0.7	1	2	4	0	1.5	0	45.04(1.95)	54.36(23.05)	52.12(17.97)	44.18
14	0.7	1	2	4	0	0	1.5	46.58(2.70)	55.86(23.15)	53.62(18.21)	45.36
15	0.5	5	3	2	1	1	1	18.51(1.61)	24.36(33.69)	20.67(13.46)	18.22
16	0.5	5	3	2	0	0	0	11.96(0.11)	19.69(64.73)	14.94(25.02)	11.95
17	0.5	5	3	2	0.5	0.5	0.5	14.79(0.08)	21.91(48.26)	17.70(19.78)	14.78
18	0.5	5	3	2	2	2	2	25.54(0.59)	29.80(17.38)	27.11(6.77)	25.39
19	0.5	5	3	2	1.5	0	0	13.58(0.19)	20.72(52.92)	16.51(21.85)	13.55
20	0.5	5	3	2	0	1.5	0	15.08(0.08)	22.22(47.45)	18.01(19.51)	15.07
21	0.5	5	3	2	0	0	1.5	15.84(0.12)	22.97(45.20)	18.76(18.58)	15.82

Table 2: Results for 3 Operations Cases, $h_1 = 10$, $h_2 = 20$, $h_3 = 30$

stations. In addition, to observe how much the "batch splitting" procedure in the heuristic is contributing to the overall performance, we report the results for the best "K-limited" policy found using simulation. Under the K-limited policy, the server processes at most K jobs in queue 1 and then switches to queue 2 (If queue 1 is exhausted before K jobs are processed, the server still switches to queue 2). The batch is never split in stations $2, 3, \ldots, N$. In cases where the percentage difference is zero the heuristic has no splitting and finds the same K as the optimal K found using simulation.

We chose three different scenarios for the mean service times. In the first case, they were identical for all stations. In the second, they were increasing in station index while in the third they were decreasing in station index. We also looked at six scenarios for setup times. These ranged from the case where all setup times were equal to 0, to the case where they were decreasing in station index and the case where they were increasing in station index. This resulted in 18 different cases. The holding costs in all 18 cases

were taken to be 10 at station 1, 20 at station 2, 30 at station 3, 40 at station 4, and 50 at station 5.

ρ	b_1, b_2, b_3, b_4, b_5	s_1, s_2, s_3, s_4, s_5	heur.	exhaustive (% dif.)	gated (% dif.)	K-limited (% dif.)
0.8	1, 1, 1, 1, 1	0, 0, 0, 0, 0	43.36	97.17(124.12)	84.16(94.11)	43.36(0.00)
0.8	1, 1, 1, 1, 1	0.5, 0.5, 0, 0, 0	71.95	118.62 (64.62)	107.20 (48.77)	80.11(10.18)
0.8	1, 1, 1, 1, 1	1, 1, 1, 1, 1	207.38	226.89 (9.41)	220.55 (6.35)	206.79 (-0.28)
0.8	1, 1, 1, 1, 1	2, 2, 0.5, 0, 0	159.40	204.32 (28.20)	197.22 (23.75)	182.57 (12.69)
0.8	1, 1, 1, 1, 1	0, 0.25,0.5, 0.75,1	137.44	165.28 (20.26)	155.96 (13.48)	137.44 (0.00)
0.8	1, 1, 1, 1, 1	1, 0.5, 0, 0, 0	80.65	129.48 (60.37)	118.78 (47.12)	95.73(15.75)
0.9	1, 1.5, 2, 2.5, 3	0, 0, 0, 0, 0	81.03	218.55 (169.71)	203.32 (150.90)	81.03(0.00)
0.9	1, 1.5, 2, 2.5, 3	0.5, 0.5, 0, 0, 0	124.90	247.26 (98.63)	232.94 (87.13)	146.02 (14.46)
0.9	1, 1.5, 2, 2.5, 3	1, 1, 1, 1, 1	316.52	374.88 (18.44)	363.91 (14.97)	311.70 (-1.54)
0.9	1, 1.5, 2, 2.5, 3	2, 2, 0.5, 0, 0	240.00	353.72 (46.80)	342.30 (42.06)	287.97 (16.66)
0.9	1, 1.5, 2, 2.5, 3	0, 0.25,0.5, 0.75,1	218.33	298.37 (36.66)	285.37 (30.71)	218.33 (0.00)
0.9	1, 1.5, 2, 2.5, 3	1, 0.5, 0, 0, 0	138.05	261.62 (91.11)	247.75 (80.98)	171.30 (19.41)
0.7	4, 3, 2, 1, 1	0, 0, 0, 0, 0	26.27	56.28(114.29)	43.07(63.99)	26.27(0.00)
0.7	4, 3, 2, 1, 1	0.5, 0.5, 0, 0, 0	31.81	60.27(90.54)	47.81(51.16)	31.81(0.00)
0.7	4, 3, 2, 1, 1	1, 1, 1, 1, 1	66.35	84.06(22.26)	74.36(8.16)	66.35(0.00)
0.7	4, 3, 2, 1, 1	2, 2, 0.5, 0, 0	55.27	76.98(34.56)	66.92(16.98)	58.85(6.08)
0.7	4, 3, 2, 1, 1	0, 0.25,0.5, 0.75,1	49.76	71.49(32.65)	60.12(11.56)	49.76(0.00)
0.7	4, 3, 2, 1, 1	1, 0.5, 0, 0, 0	35.58	62.09(75.30)	50.00(41.16)	35.58(0.00)

Table 3: Results for 5 Operation Cases, $h_1 = 10$, $h_2 = 20$, $h_3 = 30$, $h_4 = 40$, $h_5 = 50$

The percentage difference between the heuristic suggested and the exhaustive, gated and K-limited policies is given for each example in Table 3. The results in Table 3 clearly show that the heuristic dramatically outperforms the exhaustive and gated policies. The average percentage difference between the heuristic and the exhaustive polling policy was 63% while the average difference between the heuristic and the gated policy was 46%. Table 3 also shows that the heuristic can significantly outperform the K-limited policy. Although on average the percentage difference between the heuristic and the K-limited policy was only 5.2%, this is because for many of the examples there was no advantage to splitting the original batch in stations $2, \ldots, N$. However, the examples in Table 3 show that when splitting is beneficial, the difference can be high.

Notice also that, in all but two cases, when splitting did not occur the heuristic chose the optimal value of K for the K-limited policy. In the two cases where the optimal value of K is different from the heuristic's value the difference is less than the percentage error in the simulation. Also, finding K through our heuristic takes only a few seconds whereas finding K through simulation can be very tedious.

In many manufacturing applications there is one production step that adds significant value to the product. Table 4 shows examples where such a jump in holding costs is present. It also investigates the splitting procedure further and leads to added insight on our heuristic. We use a five station model with ρ=0.8 and all service times equal to 1. Here we only compare the heuristic to the optimal K-limited policy. In all cases the heuristic did the same or better than the K-limited policy so it would clearly also outperform exhaustive and gated service. Also shown in the table are the values of K and y (splitting procedure) found by the heuristic.

	h_1, h_2, h_3, h_4, h_5	s_1, s_2, s_3, s_4, s_5	heur.	K	y_2, y_3, y_4	K-limited (% dif.)
1	1, 1.2, 1.4, 1.6, 10.8	1, 1, 1, 1, 1	20.71	10	1,1,2	21.81(5.03)
2	1, 1.2, 1.4, 1.6, 20.8	1, 1, 1, 1, 1	28.82	9	1,1,2	32.59(11.57)
3	1, 1.2, 1.4, 1.6, 50.8	1, 1, 1, 1, 1	48.65	12	1,1,4	62.61(22.29)
4	1, 1.2, 1.4, 10.6, 10.8	1, 1, 1, 1, 1	29.73	10	1,2,1	31.50(5.62)
5	1, 1.2, 10.4, 10.6, 10.8	1, 1, 1, 1, 1	40.17	10	2,1,1	40.62(1.11)
6	1, 10.2, 10.4, 10.6, 10.8	1, 1, 1, 1, 1	49.63	6	1,1,1	49.63(0.00)
7	1, 1.2, 1.4, 10.6, 10.8	1, 1, 1, 0.1, 0.1	17.43	8	1,3,1	21.35(18.37)
8	1, 1.2, 1.4, 10.6, 10.8	1, 1, 0.1, 0.1, 0.1	11.12	6	1,6,1	17.39(36.06)
9	1, 1.2, 1.4, 10.6, 10.8	1, 0.1, 0.1, 0.1, 0.1	9.35	5	2,3,1	10.93(14.49)
10	1, 1.2, 1.4, 10.6, 10.8	0.1, 0.1, 0.1, 0.1, 0.1	8.16	1	1,1,1	8.16 (0.00)

Table 4: Results for 5 Operation Cases, $b_1 = 1$, $b_2 = 1$, $b_3 = 1$, $b_4 = 1$, $b_5 = 1$, and ρ=0.8

Examples 1-3 show the effect of an increase in the cost at the expensive step. Clearly, the benefits obtained from splitting become greater and the difference between the heuristic and K-limited increases with the increase in cost. Notice also that the amount of splitting increases with cost. Examples 1, 4, 5 and 6 show the effect of moving the expensive step towards the front of the line. The point at which a batch is split is moved forward until, in the final example, whole batches are sent through. Examples 7-10 show the effect of a decrease in setup time on the system. As would be expected, the system is more likely to split when setup time goes down. Also, as the point where the setup time decrease occurs moves towards the front of the line, so does the point at which splitting occurs. Example 9 is particularly noteworthy because here the splitting procedure causes a batch to be split once when setup time decreases and again when holding costs increase.

7 Conclusions and Further Research

In this paper, we presented the problem of scheduling a tandem queueing system with setups and a single-server. We partially characterized the optimal policy and provided a performance analysis of exhaustive and gated polling policies for such systems. We also developed an effective heuristic. Further research should focus on more complicated systems than the one we considered here (e.g., multiple servers, or more complex routing schemes).

Acknowledgments:

The authors are very grateful to the area editor, Professor Larry Wein, and to an anonymous associate editor and two anonymous referees whose insightful comments greatly improved the paper. Izak Duenyas has been supported in part by NSF Grants, DMI-9308290 and DMI-9424596 and a grant from the Center for Display Manufacturing. Diwakar Gupta has been supported in part by the Natural Sciences and Engineering Research Council of Canada through research grant number OGP0045904. The authors are grateful to Mr. Yavuz Gunalay, a Ph. D. candidate at McMaster University, for his help in computer implementation of algorithms based on the analysis of exhaustive and gated polling policies.

Appendix: Proof of Theorem 2

Using the data transformation suggested in Tijms [34], (page 209-210), the Semi-Markov decision model (3.4) can be converted into a discrete time Markov decision policy such that for each stationary policy, the average costs per unit time are the same in both models. This enables us to use a value iteration algorithm to solve the original Semi-Markov decision model. We let $\varphi = \max\{\psi_1, \psi_2, \lambda, \beta_1, \beta_2\}$ and we get the following recursive equation for the value iteration algorithm.

$$
\begin{aligned}
V_{k+1}(X_1, X_2, 1) &= \min \left\{ \begin{array}{l} H_1(X_1, X_2) + \frac{\beta_1}{\varphi} EV_k(X_1 + A_1(B_1) - 1, X_2 + 1, 1) + (1 - \frac{\beta_1}{\varphi}) V_k(X_1, X_2, 1) \\ H_2(X_1, X_2) + \frac{\psi_2}{\varphi} EV_k(X_1 + A_1(S_2), X_2, 2)] + (1 - \frac{\psi_2}{\varphi}) V_k(X_1, X_2, 1) \end{array} \right\} \text{ for } X_1 \geq 1 \\
V_{k+1}(X_1, X_2, 2) &= H_3(X_1, X_2) + \frac{\beta_2}{\varphi} EV_k(X_1 + A_1(B_2), X_2 - 1, 2)] + (1 - \frac{\beta_2}{\varphi}) V_k(X_1, X_2, 2) \quad \text{for } X_2 \geq 1. \\
V_{k+1}(0, X_2, 1) &= \min \left\{ \begin{array}{l} (h_2 X_2) + \frac{\lambda}{\varphi} V_k(1, X_2, 1)] + (1 - \frac{\lambda}{\varphi}) V_k(0, X_2, 1) \\ \psi_2 E[\int_0^{S_2} [h_1(A_1(t)) + h_2 X_2] dt] + \frac{\psi_2}{\varphi} EV_k(A_1(S_2), X_2, 2) + (1 - \frac{\psi_2}{\varphi}) V_k(0, X_2, 1) \end{array} \right\} \\
V(X_1, 0, 2) &= \psi_1 E[\int_0^{S_1} [h_1(X_1 + A_1(t)) dt] + \frac{\psi_1}{\varphi} EV_k(X_1 + A_1(S_1), 0, 1)] + (1 - \frac{\psi_1}{\varphi}) V_k(X_1, 0, 2)
\end{aligned}
\tag{7.49}
$$

where $H_1(X_1, X_2) = \beta_1 E[\int_0^{B_1} [h_1(X_1 + A_1(t)) + h_2 X_2] dt]$, $H_2(X_1, X_2) = \psi_2 E[\int_0^{S_2} [h_1(X_1 + A_1(t)) + h_2 X_2] dt]$, and $H_3(X_1, X_2) = \psi_2 E[\int_0^{B_2} [h_1(X_1 + A_1(t)) + h_2 X_2] dt]$ with $V_0(X_1, X_2, 1) = 0$, and $V_0(X_1, X_2, 2) = 0$ for all X_1 and X_2.

Now, in order to prove Theorem 2, it is sufficient to show that at each iteration of this value iteration algorithm, the following conditions hold true:

C1) $Z_k(X_1, X_2) = \frac{\beta_1}{\varphi} EV_k(X_1 + A_1(B_1) - 1, X_2 + 1, 1) + \frac{(\psi_2 - \beta_1)}{\varphi} V_k(X_1, X_2, 1) - \frac{\psi_2}{\varphi} EV_k(X_1 + A_1(S_2), X_2, 2)$ is increasing in X_2 and decreasing in X_1.

C2) $G_k(X_2) = \frac{\lambda}{\varphi} V_k(1, X_2, 1) + \frac{(\psi_2 - \lambda)}{\varphi} V_k(0, X_2, 1) - \frac{\psi_2}{\varphi} EV_k(A_1(S_2), X_2, 2)$ is increasing in X_2.

To see this, note for example that if at a given iteration, in a given state $(X_1, X_2, 1)$, the optimal choice is to switch to queue 2 (rather than serve another unit from queue 1) then by the first equation in (7.49), this implies that

$$H_1(X_1, X_2) - H_2(X_1, X_2) + Z_k(X_1, X_2) > 0. \tag{7.50}$$

Now note that the difference between the first two terms of (7.50) is a constant in X_1 and X_2. Therefore, if Z_k is increasing in X_2, this would imply that in state $(X_1, X_2 + 1, 1)$, it would also be optimal to switch to queue 2. Similarly, if in state $(X_1, X_2, 1)$, it is optimal to serve queue 1, it will also be optimal to serve queue 1 in state $(X_1 + 1, X_2, 1)$. The explanation for why monotonicity of $G_k(X_2)$ is required is similar.

We will now show by induction the monotonicity properties of Z_k (the proof of the monotonicity of G_k is entirely similar and omitted). In addition, we will also need the following condition (which we will also prove),

C3) $W_k(X_1, X_2) = \frac{\beta_1}{\varphi} EV_k(X_1 + A_1(B_1) - 1, X_2 + 1, 2) - \frac{\beta_2}{\varphi} EV_k(X_1 + A_1(B_2), X_2 - 1, 2) + \frac{(\beta_2 - \beta_1)}{\varphi} V_k(X_1, X_2, 2)$ is increasing in X_2 and decreasing in X_1.

Now, we first note that condition (C1) on Z_k and condition (C3) on W_k hold trivially when $k = 0$. Assume they hold for arbitrary k. Also let $(X_1, X_2)^T = (X_1 + A_1(B_1) - 1, X_2 + 1)$. Then for the $k+1^{st}$ iteration, after a lot of algebra, we can write,

$$\begin{aligned} Z_{k+1}(X_1, X_2) = \; & \tfrac{\psi_2}{\varphi} E[W_k(X_1 + A_1(S_2), X_2)] + \tfrac{\beta_1}{\varphi} E[\min\{0, H_1(X_1, X_2)^T - H_2(X_1, X_2)^T + Z_k(X_1, X_2)^T\}] \\ & + \tfrac{\psi_2 - \beta_1}{\varphi} E[\min\{0, H_1(X_1, X_2) - H_2(X_1, X_2) + Z_k(X_1, X_2)\}] + (1 - \tfrac{\psi_2}{\varphi})(Z_k(X_1, X_2) + H_1(X_1, X_2) - H_2(X_1, X_2)) \\ & + (1 - \psi_2/\varphi)(H_2(X_1, X_2) - H_1(X_1, X_2)) + (\beta_1/\varphi) EH_2(X_1, X_2)^T + \tfrac{\psi_2 - \beta_1}{\varphi} H_2(X_1, X_2) - (\psi_2/\varphi) EH_3(X_1 + A_1(S_2), X_2) \end{aligned} \tag{7.51}$$

The first term in the first line of (7.51) is increasing in X_2 and decreasing in X_1 by the induction assumption (C3) on the W terms. The second term on the first line is increasing in X_2 and decreasing in X_1 by the induction assumption (C1) on the Z terms. Similarly, the sum of the terms in the second line is increasing in X_2 and decreasing in X_1 by the induction assumption (C1) on the Z terms. Finally, the sum of the terms in the third line are constant in X_1 and X_2 from the definition of H_1, H_2 and H_3. We have therefore shown that $Z_{k+1}(X_1, X_2)$ is also increasing in X_2 and decreasing in X_1.

Similarly, after some algebra, for $X_2 > 1$, we can write,

$$W_{k+1}(X_1, X_2) = (\psi_2/\varphi)EW_k(X_1 + A_1(B_2), X_2 - 1) + (1 - \psi_2/\varphi)W_k(X_1, X_2) + \frac{\beta_1}{\varphi}EH_3(X_1, X_2)^T - \frac{\beta_2}{\varphi}EH_3(X_1 + A_1(S_2) + A_1(B_2), X_2 - 1) + \frac{(\beta_2 - \beta_1)}{\varphi}EH_3(X_1 + A_1(S_2), X_2) \quad (7.52)$$

The terms in the first line of (7.52) are increasing in X_2 and decreasing in X_1 by the induction assumption (C3) on the W terms, while the sum of the terms in the second line are constant in X_1 and X_2. The boundary case with $X_2 = 1$ is similar.

We have therefore shown that the properties of the value function that are required for the result hold true through every iteration of the value iteration algorithm. Since $V_k(X_1, X_2) \rightarrow V(X_1, X_2)$ as $k \rightarrow \infty$, $Z_k(X_1, X_2) \rightarrow Z(X_1, X_2)$ and $G_k(X_1, X_2) \rightarrow G(X_1, X_2)$ and therefore the properties hold true for the optimality equation for the average cost case, and we have therefore shown the result. □.

Bibliography

[1] Altman, E., P. Konstantopoulos, and Z. Liu, 1992, "Stability, Monotonicity and Invariant Quantities in General Polling Systems," *Queueing Systems: Theory and Applications,* **11**, 35-57.

[2] Boxma, O.J., (1991) "Analysis and Optimization of polling systems," In *Queueing, Performance and Control in ATM, (ITC-13),* (Edited by J.W. Cohen and C.D. Pack), pp.173-183, North Holland, Amsterdam.

[3] Boxma, O.J., H. Levy, and J.E. Westrate, 1990, "Optimization of Polling Systems," In *Performance '90,* (Edited by P.J.B. King, I. Mitrani, and R.J. Pooley), pp.349-361, North Holland, Amsterdam.

[4] Browne, S., and U. Yechiali, (1989), "Dynamic priority rules for cyclic-type queues", *Advances in Applied Probability,* **21**, 432-450.

[5] Cooper, R. B., (1970), "Queues Served in Cyclic Order: Waiting Times," *The Bell System Technical Journal,* **49**, 399-413.

[6] Duenyas, I., and Van Oyen, M., (1995), "Stochastic Scheduling of Parallel Queues with Set-Up Costs," *Queueing Systems: Theory and Applications,* **19** No:3, 421-444.

[7] Duenyas, I., and Van Oyen., M., (1996), "Heuristic Scheduling of Parallel Heterogeneous Queues with Set-Ups," *Management Science,* **42** 814-829.

[8] Eisenberg, M. (1994) "The Polling System with a Stopping Server, *Queueing Systems: Theory and Applications,* **18**, 387-431.

[9] Fuhrmann, S. W. and Cooper R. B. (1985). "Stochastic Decompositions in the M/G/1 Queue with Generalized Vacations." *Operations Research,* **33** (5), 1117-1129.

[10] Gupta, D. and Srinivasan, M. M. (1996), "Polling Systems with State Dependent Setup Times," forthcoming in *Queueing Systems: Theory and Applications.*

[11] Gupta, D., Y. Gerchak, and J.A. Buzacott, (1987), "On optimal priority rules for queues with switchover costs," Preprint, Department of Management Sciences, University of Waterloo.

[12] Hofri, M., and K.W. Ross, (1987) "On the optimal control of two queues with server set-up times and its analysis," *SIAM Journal of Computing,* **16**, 399-420.

[13] Iravani, S.M.R., M.J.M. Posner, and J.A. Buzacott, (1995), "Two-Stage Tandem Queue Attended by a Moving Server with Holding and Switching Costs; Static and Semi-Dynamic Policies," Technical Report, 95-18, University of Toronto, Department of Industrial Engineering, Toronto, Canada.

[14] Johri, P.K., and M. N. Katehakis (1988), "Scheduling Service in Tandem Queues Attended by a Single Server," *Stochastic Analysis and Applications,* **6** No:3, 279-288.

[15] Katayama, T., (1980) "Analysis of an Exhaustive Service Type Tandem Queue Attended by a Moving Server with Walking Time," *Transactions of IECE of Japan,* **J-63-B** No:11, 1055-1062 (in Japanese).

[16] Katayama, T., (1981) "Analysis of a Tandem Queueing System with Gate Attended by a Moving Server with Walking Time," *Transactions of IECE of Japan,* **J-64-B** No:9, 931-938 (in Japanese).

[17] Katayama, T., (1981), "Analysis of a Finite Intermediate Waiting Room Tandem Queue Attended by a Moving Server with Walking Time," *Transactions of IECE of Japan,* **E64** No:9, 571-578.

[18] Katayama, T., (1983), "Analysis of a Limiting Service Tandem Queue Attended by a Moving Server with Walking Time," *Review of the Electrical Communication Laboratory,* **31** No:3, 439-446.

[19] Katayama, T., (1988), "Mean Sojourn Times in a Multi-Stage Tandem Queue Served by a Single Server," *Journal of the Operations Research Society of Japan,* **31**, No:2, 233-247.

[20] Katayama, T., (1992), "Performance Analysis and Optimization of a Cyclic-Service Tandem Queueing System with Multi-Class Customers," *Computers and Math Applications,* **24**, No:1, 25-33.

[21] Konheim, A. G., H. Levy and M. M. Srinivasan, (1994), "Descendant Set: An Efficient Approach for the Analysis of Polling Systems," *IEEE Transactions on Communications,* **42** 1245-1253.

[22] Konig, D., and Schmit, V., (1984), "Relationships between Time/Customer Stationary Characteristics of Tandem Queues Attended by a Single Server," *Journal of the Operational Research Society of Japan,* **27**, No:3, 191-204.

[23] Lippman, S.A., (1975), "Applying a new device in the optimization of exponential queueing systems," *Operations Research,* **23**. 687-710.

[24] Liu, Z., P. Nain, and D. Towsley, (1992), "On optimal polling policies," *Queueing Systems, Theory and Applications,* **11**, 59-84.

[25] Murakami, K., and Nakamura, G., (1978), "A Model for Event Handling in a Functionally Dedicated Processor and Analysis," *Transactions of IECE of Japan,* **J61-D, 7**, 465-472.

[26] Nair, S.S., (1971), "Semi-Markov analysis of two queues in series attended by a single server," *Bull. Soc. Math. Belqique,* **22**, 355-367.

[27] Netto, M.T., (1977) "Two queues in tandem attended by a single server" *Operations Research,* **25**, No:1, 140-147.

[28] Rajan, R., and R. Agrawal, (1991) "Optimal server allocation in homogeneous queueing systems with switching costs," preprint, Electrical and Computer Engineering Department, University of Wisconsin-Madison, Madison, WI, 53706.

[29] Reiman, M.I., and L.M. Wein, (1994), "Dynamic Scheduling of a Two-Class Queue with Setups," preprint, Sloan School of Management, M.I.T., Cambridge, M.A. 02139.

[30] Sidi, M., H. Levy, and S. W. Fuhrmann, (1992), "A queueing network with a single cyclically roving server," *Queueing Systems: Theory and Applications,* **11**, 121-144.

[31] Srinivasan, M. M., Niu, S. C. and Cooper R. B. (1995). "Relating Polling Models with Zero and Nonzero Switchover Times." *Queueing Systems*, **19** (1), 149-168.

[32] Srinivasan, M. M. and D. Gupta, (1996), "When Should a Roving Server be Patient?" *Management Science*, **42**, 437-451.

[33] Takagi, H., (1990). "Queueing Analysis of Polling Models: An Update," *Stochastic Analysis of Computer and Communication Systems,* H. Takagi (Editor), Elsevier Science Publishers B.V. (North Holland), 267-318.

[34] Tijms, H.C., (1986) *Stochastic Modeling and Analysis: A Computational Approach,* John Wiley and Sons, New York.

[35] Whitt, W., (1993) "Approximations for the GI/G/M Queue," *Productions and Operations Management* **2** No:2, 114-161.