Probability in the Engineering and Informational Sciences, **17**, 2003, 351–367. Printed in the U.S.A.

DYNAMIC VISIT-ORDER RULES FOR BATCH-SERVICE POLLING

JAN VAN DER WAL
Technical University of Eindhoven
Eindhoven, The Netherlands
E-mail: jan.v.d.wal@tue.nl

URI YECHIALI
Department of Statistics and OR
Tel Aviv University
Tel Aviv, Israel
E-mail: uriy@post.tau.ac.il

We explore visit-order policies in nonsymmetric polling systems with switch-in and switch-out times, where service is in batches of unlimited size. We concentrate on so-called "Hamiltonian tour" policies in which, in order to give a *fair* treatment to the various users, the server attends every nonempty queue exactly once during each round of visits (cycle). The server dynamically generates a new visit schedule at the start of each round, depending on the current state of the system (number of jobs in each queue) and on the various nonhomogeneous system parameters. We consider three service regimes, globally gated, (locally) gated, and exhaustive, and study three different performance measures: (1) minimizing the expected weighted sum of all *sojourn* times of jobs within a cycle; (2) minimizing the expected length of the *next* cycle, and (3) maximizing the expected weighted *throughput* in a cycle. For each combination of performance measure and service regime, we derive characteristics of the optimal Hamiltonian tour. Some of the resulting optimal policies are shown to be elegant index-type rules. Others are the solutions of deterministic NP-hard problems. Special cases are reduced to assignment problems with specific cost matrices. The index-type rules can further be used to construct fixed-order, cyclic-type polling tables in cases where dynamic control is not applicable.

1. INTRODUCTION

We study optimal scheduling policies in heterogeneous polling systems with switch-in and switch-out times, where service is in batches of unlimited size. We concen-

https://doi.org/10.1017/S0269964803173044 Published online by Cambridge University Press

trate on *fair* policies, called "Hamiltonian-tours," where in each new round the server must attend every nonempty queue exactly once, but is free to determine the order of visits.

Polling systems—a set of queues attended by a single server—play a central role in the modeling and analysis of various problems in communications, computer software and hardware, database systems, manufacturing, maintenance management, and road traffic control. Much of the literature deals with occasions when service is given to *one job at a time*. For a comprehensive treatment of such models, as well as a description of various applications and extensive literature, we refer to Takagi [15–19]. Several surveys dealt with applications and optimization (Levy and Sidi [12]), applications in communications (Grillo [11]), applications in computer networks (Takagi [17]), analysis and optimization (Boxma [5]), and analysis and control (Yechiali [22]). Most of the works considered "open" polling systems in which all jobs are "transient" (i.e., they arrive to, are served by, and leave the system, never to return). A few works analyzed "closed" networks with "permanent" jobs that are routed from one queue to another, but never leave the system (Altman and Yechiali [2], Dror and Yechiali [8]). Hybrid systems with both permanent and transient jobs were analyzed by Armony and Yechiali [4].

Unlimited batch-service models were considered in the literature as applications to videotex, telex, and TDMA (Time Division Multiple Access) systems [3,9,13], as well as for central database operations [20]. Ammar and Wong [3] considered a teletext system with N queues, where queue i receives a Poisson stream of requests with rate λ_i. Service times in all queues are *deterministic* (slotted, unit of time each), batches are *unlimited*, and there are *no* switching times. The service discipline is (locally) gated. Using a homogeneous linear cost function (one unit of cost for each outstanding request at the beginning of a slot) and applying the Markov decision process formulation, they showed that the policy that minimizes the mean response time is of a cyclic nature, with cycle length of $L \geq N$ slots, in which queue i is visited k_i times, $\sum_{i=1}^{N} k_i = L$. Yet, the problem of finding the *exact* length L was only partially resolved.

Liu and Nain [13] examined the (locally) gated and exhaustive regimes for the case of *zero switching times* and a homogeneous arrival process to all queues, whereas Dykeman et al. [9] indicated (after using Howard's policy-iteration algorithm) that even with equal and deterministic service requirements and with no switching times, the structure of the optimal policy *could be very complicated*. Van Oyen and Teneketzis [20] formulated a central database system and an automated guided vehicle (AGV) as a polling system with an infinite-capacity batch server and zero switching times, where the controller observes only the length of the queue at which the server is located. A review titled "Scheduling with Batching" is presented in [14]. Batches there, however, are *finite* and are defined as the "maximal set of jobs that are scheduled to be served *continuously, one at a time,* on a machine and share a setup." The concentration there is on "classification of problems as polynomially or pseudopolynomially solvable, binary, or unitary (NP-hard), or open."

Recently, Xia, Michailidis, Bambos, and Glynn [21] considered the problem of dynamic allocation of a single server with *finite* batch processing capability to a set of parallel queues. The independent arrival processes are Poisson, but with *equal* rates. The service times of batches are exponentially distributed and *identical*. *Zero* switching times are incurred by the server when moving from one queue to another. They restrict attention to the set of *nonanticipative, nonpreemptive, and nonidling* policies and show that when buffers are infinite, allocating the server to the longest queue stochastically maximizes the aggregate throughput of the system.

Our work was motivated by a tape reading problem in a system in which large amounts of information are stored on tapes. Irregularly, requests arrive for data on one of these tapes. In order to read the data, the tape has to be mounted, read, and then dismounted. If there is more than one request for a tape, they can all be read at (more or less) the same time, thus suggesting modeling as a batch service with *unlimited* batch size. Nevertheless, the models we present in this work are *general* and can be applied to all of the above-mentioned unlimited-batch service systems. Without loss of generality, we will use the "tape language" throughout the article.

The goal is to establish "fair" dynamic visit-order policies so as to optimize various performance measures. Our special fairness approach is to visit the queues in a Hamiltonian way. (See Browne and Yechiali [7].) In each new Hamiltonian cycle, only those queues are visited that are nonempty at the start of the cycle, and each such queue is visited *exactly once*. What is to be decided at the beginning of each cycle is the order in which the queues are visited. For a cycle with duration C, we define the "sojourn time of a job J within a cycle" as the total time that J resides in the system during C. We will consider the following performance measures: (1) minimizing the expected (weighted) sum of all *sojourn times* of all jobs within a cycle, (2) minimizing the expected duration of the *next cycle*, and (3) maximizing the expected weighted *throughput* within the cycle. Criterion 2 is a proper one since, given the fairness considerations leading to Hamiltonian tours and as a result of the unlimited batch service, the duration of the current cycle is *independent* of the order of visits. Thus, one can only influence the duration of the *next cycle*.

We formulate the problem as a polling system with unlimited batch size service and consider globally gated, (locally) gated, and exhaustive service regimes. Some of the optimization problems turn out to be NP-hard, some yield simple, elegant, index-type rules, and others are transformed into classical assignment problems, but with special costs involving the parameters of the problem.

The structure of the article is as follows. In Section 2, we present the model, describe the assumptions, and discuss various gating procedures. Minimizing the weighted expected *sojourn* times is treated in Section 3, where all three service regimes are considered. Section 4 deals with minimizing the expected duration of the *next* cycle, whereas Section 5 aims at maximizing the expected weighted *throughput* in a cycle. The analytical results and their implications are summarized in Section 6, and conclusions are discussed in Section 7.

2. THE MODEL

There is a set of queues (tapes), numbered 1 to N. Requests (jobs) arrive to queue l at a Poisson rate λ_l. A single server visits the queues according to some order. When a queue is visited, all jobs are served simultaneously. This service time is independent of the number of jobs served. The total visit time at queue l consists of three parts: a time R_l to reach (mount) the tape, a time B_l to read (serve) the tape if the queue is gated, or several repetitions of B_l if the service is exhaustive, and, finally, a time D_l to dismount the tape (exit the queue to some "base" position). The order in which the queues are visited is completely free.

Hamiltonian Cycle Approach

In order to give a fair service to all queues, in each cycle the server performs a Hamiltonian tour in which every *nonempty* queue is visited *exactly once*; that is, if at the beginning of the cycle, there are $n_l \geq 0$ jobs in queue l, the server visits only those queues for which n_l is *positive*. Thus, even if during the cycle, jobs arrive to a queue that was empty at the beginning of the cycle, this queue is not included in the cycle. This process then repeats itself with a new Hamiltonian tour, where the server visits only queues for which the *new* n_l are positive.

We consider several service regimes:

- Globally gated: All queues are simultaneously gated (closed) at the start of the cycle.
- (Locally) gated: A queue is gated only when the server arrives.
- Exhaustive: The server continues serving a visited queue until it becomes empty.

In the two gated regimes, jobs arriving to a queue after its gate is closed will be served only during the next cycle. The same applies in the exhaustive regime for jobs arriving after the server leaves the queue. For a gated queue l, a visit requires a time $R_l + B_l + D_l$. We define $H_l = R_l + B_l + D_l$, $h_l = \mathrm{E}[H_l]$, $h_l^{(2)} = \mathrm{E}[H_l^2]$, and $\tilde{H}_l(\alpha) = \mathrm{E}[e^{-\alpha H_l}]$. Similarly we define r_l, $r_l^{(2)}$, $\tilde{R}_l(\alpha)$, b_l, and so forth.

Without loss of generality, we assume that at the beginning of the cycle, n_l is positive for queues $l = 1, \ldots, L$ and that $n_l = 0$ for $l = L+1, \ldots, N$. Then, for the gated cases, the duration of the cycle is $\sum_{l=1}^{L} H_l$, *independent* of the order in which the first L queues are visited. Moreover, for the exhaustive case as well, since the service is in batches, the duration of the cycle is also *independent* of the order of visits.

3. MINIMIZE THE WEIGHTED EXPECTED SOJOURN TIMES

In this section, we consider the criterion of *minimizing the expected weighted sum of sojourn times of all jobs in the cycle*. (It is important to note that under batch service, minimizing sojourn times is not always equivalent to minimizing waiting times.) We will analyze the three service regimes introduced earlier.

3.1. Globally Gated

Under the *globally gated regime* (cf. Boxma, Levy, and Yechiali [6]), there is only one thing to look at: the sojourn times of the jobs present at the beginning of the cycle. This follows because the distribution of the cycle time is independent of the visit order, implying that the waiting times of those jobs that arrive during this cycle (and are not served) are not influenced by the order. A visit order (policy) is determined by a permutation $\pi = \{\pi(1), \pi(2), \ldots, \pi(l), \ldots, \pi(L)\}$ of the set $\{1,2,\ldots,L\}$, where $\pi(l)$ indicates the index of the queue which is lth to be visited during a Hamiltonian tour. In order to find the optimal order, we start with the policy π_1 for which $\pi(l) = l$. The total expected weighted sojourn times for the jobs present at the beginning of the cycle is denoted by $C(\pi_1)$. Then, with w_l denoting the weight associated with the sojourn times of jobs in queue l,

$$C(\pi_1) = \sum_{l=1}^{L} w_l n_l \left(\left(\sum_{j=1}^{l-1} h_j \right) + r_l + b_l \right).$$

THEOREM: *The optimal policy is to visit the queues according to a nonincreasing order of the index* $w_i n_i / h_i$.

PROOF: Look at the policy π_1', which has the same order as π_1 except that queues i and $i+1$ are interchanged; that is, π_1' is the order $(1,2,\ldots,i-1,i+1,i,i+2,\ldots,L)$. Now, the difference in sojourn "costs" for these two strategies is

$$C(\pi_1) - C(\pi_1') = w_{i+1} n_{i+1} h_i - w_i n_i h_{i+1}.$$

For this to be nonpositive (i.e., for strategy π_1 to be at least as good as π_1'), we must have

$$\frac{w_i n_i}{h_i} \geq \frac{w_{i+1} n_{i+1}}{h_{i+1}}.$$

Repeatedly applying the interchange argument, the optimal visit order is determined by an *index rule*: Serve the queues in a nonincreasing order of $w_i n_i / h_i$. ■

Note that if all h_i are equal and all w_i are the same, the optimal order is according to the *longest queue first* policy, which is the consequence of the batch servicing.

This result may be compared with the result of Liu and Nain [13] for the (locally) gated case, who showed that for a *fully symmetric* polling system with *zero switching times*, "the so-called Most Customers First policy (in which the server always visits the queue with the largest number of customers) minimizes, in the sense of strong stochastic ordering, the vector of the number of customers in each queue whose components are arranged in decreasing order." Note that in their treatment, the server is free of "fairness" considerations and is entitled to choose the next queue to visit when exiting a served queue, whereas in our approach, the order within each new cycle is *determined at the cycle's beginning*.

3.2. Locally Gated

Gating queue l just before R_l or just before B_l is not essentially different. We consider the case of gating just before B_l. Gating just before R_l is then a special case with $R_l' = 0$ and $B_l' = R_l + B_l$.

As indicated, in the locally gated regime (similarly to the globally gated case), the batch servicing implies that the duration of the cycle, $\sum_{l=1}^{L} H_l$, is not influenced by the order in which the queues are visited. Therefore, we can ignore queues $L+1,\dots,N$. The difference, compared to the globally gated case, is that now we also have to consider new arrivals (during the cycle) to queues 1 to L. Assume again that the visit order is according to policy $\pi_1=(1,2,\dots,L)$.

Consider queue l and define S_l to be the total sojourn time of all queue-l jobs in the cycle. Then, S_l is comprised of four terms. The first one is the *sojourn time* of the n_l jobs present at the start of the cycle. The second term is the *waiting time* of all jobs that arrive (during the cycle) before *this* queue is gated. The third term contains the *service time* of the jobs that arrive *before* gating, and the fourth term consists of the *sojourn time*, until the end of the cycle, of those jobs that arrive *after* the queue is gated.

For the second and fourth term, we use the following well-known result.

LEMMA: *Let $X_1, X_2, \dots$ be the arrival instants of a Poisson process with arrival rate λ and let Y be a nonnegative random variable independent of the process $\{X_i\}$. Let $N(Y)$ be the number of Poisson arrivals in $(0,Y)$; that is, $N(Y)=\max\{k \mid X_k<Y\}$. Then*

$$\mathrm{E}\left[\sum_{j=1}^{N(Y)}(Y-X_j)\right]=\mathrm{E}\left[\sum_{j=1}^{N(Y)}X_j\right]=\frac{\lambda}{2}\,\mathrm{E}[Y^2].$$

Combining the above described four terms and following π_1, we have

$$\begin{aligned}\mathrm{E}[S_l]=n_l\left(\sum_{j=1}^{l-1}h_j+r_l+b_l\right)+\frac{\lambda_l}{2}\,\mathrm{E}[(H_1+\cdots+H_{l-1}+R_l)^2]\\ +\lambda_l\left(\sum_{j=1}^{l-1}h_j+r_l\right)b_l+\frac{\lambda_l}{2}\,\mathrm{E}[(B_l+D_l+H_{l+1}+\cdots+H_L)^2].\end{aligned}\tag{1}$$

Thus, the total expected sojourn cost is

$$\begin{aligned}\sum_{l=1}^{L}w_l\mathrm{E}[S_l]=\sum_{l=1}^{L}w_l\Bigg\{n_l\left(\sum_{j=1}^{l-1}h_j+r_l+b_l\right)+\frac{\lambda_l}{2}\,\mathrm{E}[(H_1+\cdots+H_{l-1}+R_l)^2]\\ +\lambda_l\left(\sum_{j=1}^{l-1}h_j+r_l\right)b_l+\frac{\lambda_l}{2}\,\mathrm{E}[(B_l+D_l+H_{l+1}+\cdots+H_L)^2]\Bigg\}.\end{aligned}\tag{2}$$

The objective function (2) can be simplified by omitting all terms of the form $\sum_{l=1}^{L}\gamma_l$ which are order independent.

We first rewrite the second term in (2):

$$\begin{aligned}&\sum_{l=1}^{L}\frac{\lambda_l w_l}{2}\,\mathrm{E}[(H_1+\cdots+H_{l-1}+R_l)^2]\\ &=\sum_{l=1}^{L}\frac{\lambda_l w_l}{2}\left(r_l^{(2)}+2r_l\sum_{j=1}^{l-1}h_j+\sum_{j=1}^{l-1}h_j^{(2)}+\sum_{j=1}^{l-1}\sum_{k=1}^{l-1}h_jh_k-\sum_{j=1}^{l-1}h_j^2\right).\end{aligned}\tag{3}$$

Clearly, the first term in the right-hand side of (3) is order independent.

Now consider the fourth term in (2). We can write it as

$$\sum_{l=1}^{L} \frac{\lambda_l w_l}{2} \left\{ (b_l^{(2)} + 2b_l d_l + d_l^{(2)}) + 2(b_l + d_l) \sum_{j=l+1}^{L} h_j + \sum_{j=l+1}^{L} h_j^{(2)} + \sum_{j=l+1}^{L} \sum_{k=l+1}^{L} h_j h_k - \sum_{j=l+1}^{L} h_j^2 \right\}.$$

Thus, in (2), we can omit not only the terms $\sum_l w_l n_l (r_l + b_l)$, $\sum_l w_l \lambda_l r_l b_l$, $\sum_l \lambda_l w_l r_l^{(2)}/2$, and $\sum_l \lambda_l w_l (b_l^{(2)} + 2b_l d_l + d_l^{(2)})/2$ but also the combined terms $\sum_l \lambda_l w_l \sum_{j \neq l} h_j^{(2)}/2$ and $\sum_l \lambda_l w_l \sum_{j \neq l} h_j^2/2$.

Writing $h[k,m] = \sum_{j=k}^{m} h_j$, the goal is to find a permutation π that minimizes

$$\sum_{l=1}^{L} w_{\pi(l)}(n_{\pi(l)} + \lambda_{\pi(l)} b_{\pi(l)}) h[\pi(1), \pi(l-1)] + \sum_{l=1}^{L} w_{\pi(l)} \lambda_{\pi(l)} \left\{ r_{\pi(l)} h[\pi(1), \pi(\ell - 1)] + (b_{\pi(l)} + d_{\pi(l)}) h[\pi(l+1), \pi(L)] + \frac{1}{2} (h[\pi(1), \pi(l-1)])^2 + \frac{1}{2} (h[\pi(l+1), \pi(L)])^2 \right\}. \quad \textbf{(4)}$$

This is combinatorially a NP-hard problem, as we prove in the Appendix.

We now show that the symmetric case leads to a special assignment problem, which is solvable in $O(L^3)$ time [1].

A symmetric case. Suppose $b_l = b$, $d_l = d$, and $r_l = r$, so that $h_l = h$ (this might be the case when all tapes are similar). One can verify that the objective can be simplified to finding the permutation π that minimizes

$$\sum_{l=1}^{L} w_{\pi(l)}(n_{\pi(l)} + \lambda_{\pi(l)} b) hl + \sum_{l=1}^{L} w_{\pi(l)} \lambda_{\pi(l)} \{[2rh - (L+2)h^2] l + h^2 l^2\}$$
$$= \sum_{l=1}^{L} \alpha_{\pi(l)} l + \sum_{l=1}^{L} \beta_{\pi(l)} l^2,$$

with $\alpha_{\pi(l)} = w_{\pi(l)}(n_{\pi(\ell)} + \lambda_{\pi(\ell)} b) h + w_{\pi(l)} \lambda_{\pi(l)} [2rh - (L+2)h^2]$ and $\beta_{\pi(l)} = w_{\pi(l)} \lambda_{\pi(l)} h^2$. It follows that this problem can be formulated and readily solved as an assignment problem with costs $C_{ij} = C_{\pi(i),j} = \alpha_{\pi(i)} j + \beta_{\pi(i)} j^2$ for $i, j = 1, 2, \ldots, L$.

3.3. Exhaustive Regime

In the exhaustive regime, servicing in queue l is repeated until the queue is empty. This means that if during the service time B_l, one or more new jobs arrive (with probability $\int_0^\infty (1 - e^{-\lambda_l t})\, dP(B_l \leq t) = 1 - \tilde{B}_l(\lambda_l)$), then after completing the present service time, a new service time is started. This can be repeated several times in a geometric fashion.

Let E_l denote this geometric sum of service times in queue l, with repetition rate $1-\tilde{B}_l(\lambda_l)$, mean e_l, second moment $e_l^{(2)}$, and Laplace transform $\tilde{E}_l(\alpha)$. Then,

$$e_l = \frac{b_l}{\tilde{B}_l(\lambda_l)}.$$

The Laplace transform $\tilde{E}_l(\alpha)$ is derived as follows.

For given values $B_{l1}, B_{l2}, \ldots$, all distributed as B_l, we have $E_l = \sum_{j=1}^{k} B_{lj}$, with probability $\prod_{j=1}^{k-1}(1-e^{-\lambda_l B_{lj}})e^{-\lambda_l B_{lk}}$, $k = 1,2,\ldots$. Thus,

$$\begin{aligned}\tilde{E}_l(\alpha) &= \sum_{k=1}^{\infty} E\left[e^{-\alpha\left(\sum_{j=1}^{k} B_{lj}\right)} \prod_{j=1}^{k-1}(1-e^{-\lambda_l B_{lj}})e^{-\lambda_l B_{lk}}\right]\\ &= \tilde{B}_l(\alpha+\lambda_l)\sum_{k=1}^{\infty}[\tilde{B}_l(\alpha)-\tilde{B}_l(\alpha+\lambda_l)]^{k-1}\\ &= \frac{\tilde{B}_l(\alpha+\lambda_l)}{1-\tilde{B}_l(\alpha)+\tilde{B}_l(\alpha+\lambda_l)}.\end{aligned}$$

By differentiation, we get

$$e_l^{(2)} = \frac{1}{\tilde{B}_l(\lambda_l)}\left(b_l^{(2)} + 2b_l e_l + 2e_l \tilde{B}_l'(\lambda_l)\right).$$

Let G_l be the total visit time to queue l: $G_l = R_l + E_l + D_l$. Then, for the weighted sojourn time criterion, there is only a slight difference between the gated and the exhaustive cases. If in the analysis of the gated model, we replace H_l, h_l, and $h_l^{(2)}$ by G_l, g_l, and $g_l^{(2)}$, respectively, and we add the term for the sojourn times of the jobs served during the repetitions (on the average, $\lambda_l e_l$ jobs, each requiring one residual and one normal service time), then for policy π_1, we get (cf. (2))

$$\begin{aligned}\sum_{l=1}^{L} w_l \,\mathrm{E}[S_l] = \sum_{l=1}^{L} w_l \Bigg\{ & n_l\left(\sum_{j=1}^{l-1} g_j + r_l + b_l\right) + \frac{\lambda_l}{2}\mathrm{E}\left[\left(\sum_{j=1}^{l-1} G_j + R_l\right)^2\right]\\ & + \lambda_l\left(\sum_{j=1}^{l-1} g_j + r_l\right) b_l + \lambda_l e_l\left(\frac{b_l^{(2)}}{2b_l} + b_l\right)\\ & + \frac{\lambda_l}{2}\mathrm{E}\left[\left(D_l + \sum_{j=l+1}^{L} G_j\right)^2\right]\Bigg\}. \qquad \textbf{(5)}\end{aligned}$$

The structure of this expression looks similar to (2), but, in fact, it is even more involved than in the (locally) gated case. Even for the symmetric case, when all R_l, B_l, and D_l are the same, the $\tilde{B}_l(\lambda_l)$ terms cause us to no longer have an assignment problem as a solution of (5).

3.4. Extension: Modifying the Hamiltonian Tour

A possibly improved procedure (within the framework of Hamiltonian tours) is the following. Suppose the visit order dictated by the performance measure and the gating procedure is $1,2,\ldots,L$. Then, after exiting queue 1, the rule can be reapplied with regard to the *remaining* $L-1$ queues, taking into account the new values of the n_l's. Accordingly, the next queue to be visited will be determined, continuing in this manner until the last queue (among the original L) is serviced. Note, however, that such a modified procedure will violate the main purpose of the globally gated regime aimed at *not* letting jobs arriving during the current cycle to be serviced *before* jobs that have arrived during the previous cycle. Considering for a moment a Markov decision process approach, which aims at acting optimally at each step, one can apply a new visit-order rule (e.g., the nonincreasing $w_l n_l/h_l$ policy) after each visit of a queue, doing it in a continuous routine while neglecting the Hamiltonian tour restriction and generating a dynamic, although erratic, polling visit table (for more about polling tables, see Yechiali [22]).

4. MINIMIZE THE NEXT CYCLE DURATION

Since in all three service regimes the duration of each Hamiltonian cycle is independent of the order of visits, a criterion which seems to look a bit further ahead is to *minimize the expected duration of the next Hamiltonian cycle*. In addition to short cycles being clearly good, this criterion also implies that we are trying to serve new arrivals already in the first cycle.

A possibly different way of looking one cycle ahead is to *minimize the expected number of nonempty queues at the beginning of the next cycle*. These two problems are, as we will see, closely related.

Consider again the strategy π_1 which serves the queues in the order $1,2,\ldots,L$. For this strategy, we will compute the expected duration of the next cycle. In order to do so, we need the probability that a specific queue will be empty in the beginning of the next cycle. Define

$$p_l = \text{Prob}\{\text{Queue } l \text{ is empty at the start of the next cycle}\}.$$

We emphasize again that for the globally and locally gated regimes, as well as for the exhaustive case, the duration of the present Hamiltonian cycle does not depend on the visit order within the cycle. First, consider queues not visited in this cycle. The probability that queue l, $l > L$, is empty in the beginning of the next cycle is independent of the order. Formally, defining for the gated regimes $H^{(L)} = \sum_{j=1}^{L} H_j$, we have

$$p_l = \text{E}[e^{-\lambda_l H^{(L)}}] = \tilde{H}^{(L)}(\lambda_l), \qquad l > L.$$

In the exhaustive regime, we get a similar expression with $G^{(L)} = \sum_{l=1}^{L} G_l$.

4.1. Globally Gated

When simultaneously gating at the beginning of the cycle, the p_l for $l = 1, \ldots, L$ also satisfy $p_l = \tilde{H}^{(L)}(\lambda_l)$. Hence, in this case, all orders are stochastically identical (and optimal).

4.2. Locally Gated

Similar to the sojourn times criterion, the gating is just before B_l. Then, for $l \leq L$, p_l is the probability that there are no arrivals into queue l during the remaining duration of the present cycle $B_l + D_l + H_{l+1} + \cdots + H_L$. Hence,

$$p_l = \tilde{B}_l(\lambda_l)\tilde{D}_l(\lambda_l) \prod_{j=l+1}^{L} \tilde{H}_j(\lambda_l).$$

The expected duration of the next cycle under π_1 is

$$\sum_{l=1}^{N}(1-p_l)h_l = \sum_{l=1}^{N} h_l - \sum_{l=1}^{L} \tilde{B}_l(\lambda_l)\tilde{D}_l(\lambda_l)h_l \prod_{j=l+1}^{L} \tilde{H}_j(\lambda_l) - \sum_{l=L+1}^{N} \tilde{H}^{(L)}(\lambda_l)h_l. \quad \textbf{(6)}$$

We immediately observe that, in contrast to the "minimizing the sojourn time" performance measure, the above expression is *independent* of the number of jobs present at the start of the cycle.

The first and third terms in (6) are order independent, so we consider

$$V(\pi_1) = \sum_{l=1}^{L} \tilde{B}_l(\lambda_l)\tilde{D}_l(\lambda_l)h_l \prod_{j=l+1}^{L} \tilde{H}_j(\lambda_l)$$

and wish to find a permutation π that maximizes

$$V(\pi) = \sum_{l=1}^{L} \tilde{B}_{\pi(l)}(\lambda_{\pi(l)})\tilde{D}_{\pi(l)}(\lambda_{\pi(l)})h_{\pi(l)} \prod_{j=l+1}^{L} \tilde{H}_{\pi(j)}(\lambda_{\pi(l)}).$$

In general, this expression is hard to maximize. Therefore, we consider two special cases:

1. All λ_l are equal, say $\lambda_l = \lambda$.
2. All H_l are identical.

4.2.1. Case 1: All λ_l are equal. Assume $\lambda_l = \lambda$ for all l. As previously, compare two strategies, π_1 and π_1', which differ only in the order in which queues i and $i+1$ are visited. In π_1, the order is i, $i+1$, and in π_1' the order is reversed.

Comparing $V(\pi_1)$ and $V(\pi_1')$, the contributions for queues l less than i and larger than $i+1$ are the same. We only have to compare the expressions for i and $i+1$ in the two orders.

It follows that π_1 is better than π_1' if, simplifying the notation to $a_j := \widetilde{H}_j(\lambda)$ and $c_l = \tilde{B}_l(\lambda)\widetilde{D}_l(\lambda)h_l$,

$$V(\pi_1) - V(\pi_1') = (c_i a_{i+1} + c_{i+1} - c_{i+1} a_i - c_i) \prod_{l=i+2}^{L} a_l > 0;$$

that is, if

$$\frac{c_{i+1}}{1-a_{i+1}} - \frac{c_i}{1-a_i} > 0.$$

Therefore, if all arrival rates are equal, the optimal visit schedule follows a nondecreasing order of the index

$$\frac{c_l}{1-a_l} = \frac{\tilde{B}_l(\lambda)\widetilde{D}_l(\lambda)h_l}{1-\widetilde{H}_l(\lambda)}.$$

4.2.2. Case 2: All R_l, B_l, and D_l are identical. If all R_l, B_l, D_l, and, therefore, H_l are identical, writing $f_l = \tilde{B}_l(\lambda_l)\widetilde{D}_l(\lambda_l)h$ and $g_l = \widetilde{H}(\lambda_l)$, the expression for $V(\pi_1)$ simplifies to

$$V(\pi_1) = \sum_{l=1}^{L} f_l g_l^{L-l}.$$

Thus, similar to the consideration in Section 3, if queue i is in position j in the visit order, then its “contribution” is $f_i g_i^{L-j}$. Therefore, in order to maximize $V(\pi)$, one has to solve an assignment problem with rewards $C_{ij} = C_{\pi(i),j} = f_{\pi(i)} g_{\pi(i)}^{L-j}$.

Note that this case is more involved than case 1, but essentially simpler than the general case in which the terms in the objective function depend not only on $\pi(i)$ and j but also on the order of the other queues.

4.3. Exhaustive Regime

The time to visit a queue now consists of $G_l = R_l + E_l + D_l$ instead of $H_l = R_l + B_l + D_l$. Due to the exhaustive discipline (and under π_1), we have $p_l = \widetilde{D}_l(\lambda_l)\prod_{j=l+1}^{L}\widetilde{H}_j(\lambda_l)$ for $l \le L$. Thus, writing $g_l = r_l + e_l + d_l$, the expression equivalent to $V(\pi_1)$ in Section 4.2 is

$$\sum_{l=1}^{L} \widetilde{D}_l(\lambda_l)(r_l + e_l + d_l) \prod_{j=l+1}^{L} \widetilde{R}_j(\lambda_l)\widetilde{E}_j(\lambda_l)\widetilde{D}_j(\lambda_l).$$

This leads to essentially the same optimization problem as for the locally gated variant.

The case where all λ_l are equal is the same as earlier, implying that the optimal visit schedule is determined by the nondecreasing order of

$$\frac{\tilde{D}_l(\lambda)g_l}{1-\tilde{H}_l(\lambda)} = \frac{\tilde{D}_l(\lambda)(r_l+e_l+d_l)}{1-\tilde{R}_l(\lambda)\tilde{E}_l(\lambda)\tilde{D}_l(\lambda)}.$$

However, for case 2 (all H_l equal), there is a significant difference in the meaning of the assumption that $R_l + E_l + D_l$ is the same for all l. It implicitly implies that if the H_l are the same and the E_l are the same, then all λ_l are the same as well. Hence, in this case, all queues are stochastically identical and all visit orders are optimal.

4.4. Number of Nonempty Queues

As was indicated, the problem of minimizing the expected number of nonempty queues at the beginning of the next cycle is not very different from the problem of minimizing the expected duration of the next cycle. The only difference is that in the expression for the objective function given in (6), the terms h_l disappear because the expected number of nonempty queues equals $\sum_{l=1}^{N}(1-p_l)$. Thus, the structure of the optimization problem is exactly the same.

5. MAXIMIZE THE WEIGHTED THROUGHPUT IN A CYCLE

A third objective for optimization is to maximize the expected weighted throughput during the cycle. This objective may seem to be more interesting from the viewpoint of the system operator than from the viewpoint of the requester. However, the more jobs that are served in the current cycle, the fewer jobs are left to wait until the next cycle. Thus, this objective is an interesting one for the requesters as well. For the globally gated case, the cycle throughput is order independent. Also, as we will see, the structure of the optimization problem is the same for the locally gated and for the exhaustive regimes.

Note, again, that for all service disciplines, the duration of the cycle is independent of the order in which the queues are visited, implying that one can ignore the queues $L+1$ up to N.

5.1. Locally Gated

Let π_1 be again the order $1, 2, \ldots, L$. Define $M(\pi_1)$ to be the expected weighted throughput during the cycle under order π_1. Then, gating just before B_l,

$$M(\pi_1) = \sum_{l=1}^{L} w_l n_l + \sum_{l=1}^{L} w_l \lambda_l \left(\sum_{j=1}^{l-1} h_j + r_l \right).$$

Now, similar to what we have seen previously, the optimal visit order is an index rule and is independent of n_l: Visit the queues in a nondecreasing order of

$$\frac{w_l \lambda_l}{h_l}.$$

Note that this rule does not affect the long-run weighted throughput, which is fixed and equal to $\sum_l w_l \lambda_l$, but focuses on the *early* weighted throughput.

Remark: If all w_l are the same and all λ_l are equal, then the optimal order is in a nonincreasing order of h_l, enabling accumulation of as many jobs as possible in queues to be visited later in the cycle. Similarly, if all w_l and h_l are equal, the queues are visited in a nondecreasing order of λ_l, again in order to generate as many jobs as possible in the most active queues.

5.2. Exhaustive Regime

The result here is very similar to the one derived for the locally gated case. The corresponding expression for $M(\pi_1)$ changes into

$$M(\pi_1) = \sum_{l=1}^{L} w_l n_l + \sum_{l=1}^{L} w_l \lambda_l \left(\sum_{j=1}^{l-1} g_j + r_l + e_l \right).$$

Therefore, the optimal visit schedule follows a nondecreasing order of

$$\frac{w_l \lambda_l}{g_l}.$$

6. SUMMARY OF RESULTS

Table 1 summarizes the analytical results considering the optimal policies (visit-order rules) for the various combinations of performance measure and gating procedure. The criterion "minimizing the weighted expected sojourn times (during a Hamiltonian tour)" leads to rules involving the values of the n_l's (i.e., the number of requests present at the start of the cycle in each queue). This implies that the visit order will change from one cycle to another as a result of the dynamic evolution of the system. Furthermore, it may enable one to modify the visit order for the remaining queues in the cycle each time the server exits a queue. One can possibly exercise a one-step look-ahead procedure (following the relevant rule) and apply it repeatedly without being confined by the Hamiltonian tour restriction. The objectives "minimizing the expected duration of the next cycle" and "maximizing the expected weighted throughput" lead to Hamiltonian procedures that do *not* involve the n_l's. Thus, if we try to modify the visit order within a cycle, we will come up with the same visit order that has been determined at the start of the cycle. However, one can use rules which are λ_l dependent to determine static, fixed-order, cycles in cases where dynamic control is not applicable.

7. CONCLUSIONS

We have considered the problem of finding dynamic visit-order rules for a polling system with unlimited batch servicing, where service times are independent of the batch sizes. We adopt a "fair" dynamic Hamiltonian cycle approach in which a new visit order is determined in each new cycle, based on the dynamic evolution of the

Table 1. Optimal Policies for the Various Combinations of the Performance Measure and Gating Procedure

Gating Procedure Performance Measure	Globally Gating	Locally Gating	Exhaustive
Minimize weighted expected sojourn times (Sect. 3)	**Index rule:** decreasing $w_l n_l / h_l$	Hard combinatorial problem involving the n_l's **Special case:** $h_l = h$; distinct λ_l's: assignment problem with costs c_{ij}'s involving the n_l's	Hard combinatorial problem involving the n_l's
Minimize expected duration of next cycle (Sect. 4)	All policies are stochastically equal	Hard combinatorial problem *not* involving the n_l's **Special case 1:** $\lambda_l = \lambda$ **Index rule:** increasing $\frac{\tilde{B}_l(\lambda)\tilde{D}_l(\lambda)h_l}{[1-\tilde{H}_l(\lambda)]}$ **Special case 2:** $h_l = h$: assignment problem with rewards c_{ij}'s *not* involving the n_l's	Hard combinatorial problem *not* involving the n_l's **Special case 1:** $\lambda_l = \lambda$ **Index rule:** increasing $\frac{\tilde{D}_l(\lambda)g_l}{[1-\tilde{G}_l(\lambda)]}$ **Special case 2:** $h_l = h$, $e_l = e$: fully symmetric, all policies stochastically equal
Maximize expected weighted throughput in a cycle (Sect. 5)	All policies are stochastically equal	**Index rule:** increasing $w_l \lambda_l / h_l$	**Index rule:** increasing $w_l \lambda_l / g_l$

system. Within the cycle, we have considered various performance measures and various gating procedures. Some of the Hamiltonian cycle problems lead to elegant solutions in the form of an index rule. Some lead to an assignment problem, whereas others result in combinatorial hard problems. Furthermore, the n_l-dependent rules may be reapplied within a cycle to modify it or as a one-step look-ahead repeated procedure. The non-n_l-dependent index rules can be used to construct fixed-order cyclic polling tables in cases where static rules are required.

Acknowledgment

We thank Professor Arie Tamir for very helpful discussions regarding the NP-Hard property.

References

1. Ahuja, R.K., Magnanati, T.L., & Orlin, J.B. (1993). *Network flows, theory, algorithms and applications*. Englewood Cliffs, NJ: Prentice–Hall.
2. Altman, E. & Yechiali, U. (1994). Polling in a closed network. *Probability in the Engineering and Informational Sciences* 8(3): 327–343.
3. Ammar, M.H. & Wong, J.W. (1987). On the optimality of cyclic transmission in teletext systems. *IEEE Transactions on Communications* 35(1): 68–73.
4. Armony, R. & Yechiali, U. (1999). Polling systems with permanent and transient customers. *Stochastic Models* 15(3): 395–427.
5. Boxma, O.J. (1991). Analysis and optimization of polling systems. In J.W. Cohen & C.D. Pack (eds.), *Queueing, performance and control in ATM*. Amsterdam: North-Holland, pp. 173–183.
6. Boxma, O., Levy, H., & Yechiali, U. (1991). Cyclic reservation schemes for efficient operation of multiple-queue single-server systems. *Annals of Operations Research* 36: 187–208.
7. Browne, S. & Yechiali, U. (1989). Dynamic priority rules for cyclic-type queues. *Advances in Applied Probability* 21: 432–450.
8. Dror, H. & Yechiali, U. (2000). Closed polling models with failing nodes. *Queueing Systems* 35: 55–81.
9. Dykeman, H.D., Ammar, M.H., & Wong, J.W. (1986). Scheduling algorithms for videodex systems under broadcast delivery. In *Proceedings of the International Conference on Communications (ICC'86)*, pp. 1847–1851.
10. Garey, M.R. & Johnson, D.S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W.H. Freeman.
11. Grillo, D. (1990). Polling mechanism models in communications systems—Some applications and examples. In H. Takagi (ed.), *Stochastic analysis of computer and communication systems*. Amsterdam: North-Holland, pp. 659–698.
12. Levy, H. & Sidi, M. (1990). Polling systems: Applications, modeling, and optimization. *IEEE Transactions on Communications* 38: 1750–1760.
13. Liu, Z. & Nain, P. (1992). Optimal scheduling in some multiqueue single-server systems. *IEEE Transactions on Automatic Control* 37(2): 247–252.
14. Potts, C.N. & Kovalyov, M.Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research* 120: 228–249.
15. Takagi, H. (1986). *Analysis of polling systems*. Cambridge, MA: MIT Press.
16. Takagi, H. (1990). Queueing analysis of polling systems: An update. In H. Takagi (ed.), *Stochastic analysis of computer and communication systems*. Amsterdam: North-Holland, pp. 267–318.
17. Takagi, H. (1991). Application of polling models to computer networks. *Computer Networks and ISDN Systems* 22: 193–211.
18. Takagi, H. (1997). Queueing analysis of polling models: Progress in 1990–1994. In J.H. Dshalalow (ed.), *Frontiers in queueing: Models and applications in science and engineering*. Boca Raton, FL: CRC Press, pp. 119–146.

19. Takagi, H. (1998). Queueing analysis of polling models. *ACM Computing Surveys* 20: 5–28.
20. Van Oyen, M.P. & Teneketzis, D. (1996). Optimal batch service of a polling system under partial information. *Method and Models in OR* 44(3): 401–419.
21. Xia, C.H., Michailidis, G., Bambos, N., & Glynn, P.W. (2002). Optimal control of parallel queues with batch service. *Probability in the Engineering and Informational Sciences* 16(3): 289–307.
22. Yechiali, U. (1993). Analysis and control of polling systems. In L. Donatielo & R. Nelson (eds.), *Performance evaluation of computer and communication systems*. New York: Springer-Verlag, pp. 630–650.

APPENDIX
The Scheduling Problem (4) is NP-Hard

We wish to show that the scheduling-optimization problem (4) of finding a permutation π so as to

$$\textit{Minimize}\left\{\sum_{l=1}^{L}\omega_{\pi(l)}(n_{\pi(l)}+\lambda_{\pi(l)}b_{\pi(l)})\left(\sum_{j=1}^{l-1}h_{\pi(j)}\right)\right.$$
$$+\sum_{l=1}^{L}\omega_{\pi(l)}\lambda_{\pi(l)}\left[r_{\pi(l)}\left(\sum_{j=1}^{l-1}h_{\pi(j)}\right)+(b_{\pi(l)}+d_{\pi(l)})\left(\sum_{j=l+1}^{L}h_{\pi(j)}\right)\right.$$
$$\left.\left.+\frac{1}{2}\left(\sum_{j=1}^{l-1}h_{\pi(j)}\right)^2+\frac{1}{2}\left(\sum_{j=l+1}^{L}h_{\pi(j)}\right)^2\right]\right\} \qquad \textbf{(A.1)}$$

is NP-hard.

To prove this assertion, we consider a special instance of the problem where, for all l, $b_l = 0$, $d_l = r_l$, and $w_l = 1$. Then problem (A.1) is reduced to minimizing

$$f(\pi)=\sum_{l=1}^{L}n_{\pi(l)}\left(\sum_{j=1}^{l-1}h_{\pi(j)}\right)+\frac{1}{2}\sum_{l=1}^{L}\lambda_{\pi(l)}\left[\left(\sum_{j=1}^{l-1}h_{\pi(j)}\right)^2+\left(\sum_{j=l+1}^{L}h_{\pi(j)}\right)^2\right]. \qquad \textbf{(A.2)}$$

It is known [10] that the following *partition* problem is NP-complete.

Input: $a_1, a_2, \ldots, a_k$ are positive integers $(k \geq 2)$.

Output: For $\sum_{j=1}^{k} a_j = A$, is there a subset $S \subseteq \{1,2,\ldots,k\}$ such that $\sum_{j\in S} a_j = A/2$?

That is, the process of finding an answer Yes or No to the above output question is NP-hard.

We start with transforming the partition problem into the polling-optimization scheduling problem by taking $L = k+1$; $n_l = 1$ for $l = 1,\ldots,k+1$; $h_l = a_l$ for $l = 1,2,\ldots,k$; $h_{k+1} = h_L = 1$; $\lambda_l = 1$ for $l = 1,2,\ldots,k$, and $\lambda_{k+1} = \lambda_L = M = 4(k+1)A^2$.

We now prove the following.

Claim: *The output for the partition problem is Yes if and only if there exists a permutation $\bar{\pi}$ of $\{1,2,\ldots,k+1\}$ such that $f(\bar{\pi})$, the objective value of the respective scheduling-optimization problem, satisfies*

$$f(\bar{\pi}) \leq (k+1)A + \frac{k}{2}A^2 + \frac{MA^2}{4} = B, \qquad \textbf{(A.3)}$$

where $M = 4(k+1)A^2$.

PROOF: Assume that there exists a partition and construct a permutation $\bar{\pi}$ satisfying $f(\bar{\pi}) \le B$. Suppose that the set $S \subseteq \{1,2,\ldots,k\}$ is a partition; that is, $\sum_{j\in S} a_j = \sum_{j\in S} h_j = A/2$. Let $|S| = m$. Let $\bar{\pi}$ be a permutation of $\{1,2,\ldots,k+1\}$ satisfying

$$\begin{aligned}
&\bar{\pi}(j) \in S \quad \text{for } j = 1,2,\ldots,m \ (m \le k-1),\\
&\bar{\pi}(m+1) = L,\\
&\bar{\pi}(j) \in \{1,2,\ldots,k\} - S \quad \text{for } j = m+2,\ldots,k+1.
\end{aligned}$$

Substituting for $\bar{\pi}$ in $f(\cdot)$, we obtain

$$\begin{aligned}
f(\bar{\pi}) &= \sum_{l=1}^{L}\left(\sum_{j=1}^{l=1} h_{\bar{\pi}(j)}\right) + \frac{1}{2}\sum_{\substack{l=1\\ l\ne m+1}}^{L}\left[\left(\sum_{j=1}^{l-1} h_{\bar{\pi}(j)}\right)^2 + \left(\sum_{j=\ell+1}^{L} h_{\bar{\pi}(j)}\right)^2\right]\\
&\quad + \frac{1}{2}\lambda_L\left[\left(\sum_{j=1}^{m} h_{\bar{\pi}(j)}\right)^2 + \left(\sum_{j=m+2}^{L} h_{\bar{\pi}(j)}\right)^2\right]\\
&\le LA + \frac{1}{2}(L-1)\left(\sum_{j=1}^{k} h_j\right)^2 + \frac{1}{2}M\left[\left(\sum_{j\in S} a_j\right)^2 + \left(\sum_{j\notin S} a_j\right)^2\right]\\
&= LA + \frac{L-1}{2}A^2 + \frac{1}{2}M\left[\left(\frac{A}{2}\right)^2 + \left(\frac{A}{2}\right)^2\right] = LA + \frac{L-1}{2}A^2 + \frac{M}{4}A^2 = B. \qquad \textbf{(A.4)}
\end{aligned}$$

The inequalities follow since, given $h_j = a_j$ positive integers for $j = 1,2,\ldots,L-1$, and $h_L = 1$, $\sum_{j=1}^{l-1} h_{\bar{\pi}(j)} \le \sum_{j=1}^{L-1} a_j = A$ for every $l = 2,3,\ldots,L$.

Next, suppose that the answer to the output problem is No; that is, there is no partition such that $\sum_{j\in S} a_j = A/2$. We will show that for *any* permutation π, $f(\pi) > B$. Consider an arbitrary permutation π of $\{1,2,\ldots,k+1 = L\}$ and suppose that $\pi(m) = k+1 = L$ for some m. Then

$$f(\pi) \ge \frac{1}{2}\lambda_{\pi(m)}\left[\left(\sum_{j=1}^{m-1} h_{\pi(j)}\right)^2 + \left(\sum_{j=m+1}^{L} h_{\pi(j)}\right)^2\right].$$

Since there is no partition, $\sum_{j=1}^{m-1} h_{\pi(j)} \equiv x \ne A/2$ and $\sum_{j=m+1}^{L} h_{\pi(j)} \equiv y \ne A/2$, where $x + y = \sum_{j=1}^{k} a_j = A$.

Hence, since x and y are integers, $x^2 + y^2 \ge (A/2 - \frac{1}{2})^2 + (A/2 + \frac{1}{2})^2$ (this follows from the convexity and symmetry of the program $\min\{x^2 + y^2\}$ s.t. $x + y = A; x, y \ge 0$). Therefore,

$$\begin{aligned}
f(\pi) &\ge \frac{1}{2}M\left[\left(\frac{A}{2} - \frac{1}{2}\right)^2 + \left(\frac{A}{2} + \frac{1}{2}\right)^2\right]\\
&= \frac{M}{2}\left(\frac{A^2}{2} + \frac{1}{2}\right) = \frac{M}{4}A^2 + \frac{M}{4} = \frac{M}{4}A^2 + (k+1)A^2\\
&= \frac{M}{4}A^2 + \frac{k}{2}A^2 + \left(\frac{k}{2} + 1\right)A^2 > \frac{M}{4}A^2 + \frac{k}{2}A^2 + (k+1)A = B. \qquad \textbf{(A.5)}
\end{aligned}$$

The last inequality follows since, for $k \ge 2$ and A a positive integer satisfying $A \ge 2$, $(k/2 + 1)A^2 > (k+1)A$. ■

To summarize, solving our polling scheduling-optimization problem is equivalent to solving the partition problem, which is NP-hard.