A Queueing Model for Call Blending in Call Centers

Sandjai Bhulai and Ger Koole

***Abstract*—Call centers that apply call blending obtain high-productivity and high-service levels by dynamically mixing inbound and outbound traffic. We show that agents should be assigned to outbound calls if the number of available agents exceeds a certain threshold. This control policy is optimal for equal service time distributions and a very good approximation otherwise.**

***Index Terms*—Call blending, call centers, queueing model, threshold policies.**

I. Introduction

In this note, we consider a queueing system with two types of jobs. The first type of jobs has a constraint on the performance, i.e., the average waiting time has to be below a certain level. Next to this time-constrained type there is a second type of jobs, available in an infinite quantity, for which the objective is to serve as many as possible. The arrivals of the first job type are determined by a Poisson process and the service times of both job types are independent exponentially distributed.

Both job types are served by a common pool of s servers under a nonpreemptive discipline. The question that we will answer in this note is how to schedule these s servers to maximize the throughput of type 2 jobs while satisfying the waiting time constraint on the type 1 jobs. Scheduling a type 1 job and delaying a type 2 job does not change the throughput, since we focus on the long-term throughput. Therefore the question is, when a server becomes idle and there are no type 1 jobs in the queue, whether this server should start serving a type 2 job or wait for a type 1 job to arrive. The optimal decision will be a function of the state of the other servers.

A typical application of our model is call blending in call centers. Modern call centers deal with a mixture of incoming and outgoing calls. Of course, there are constraints on the waiting time of incoming calls. The traditional solution is to assign call center employees (often called "agents") to either incoming or outgoing calls. However, the call rate fluctuates over the day and in order to handle the calls during peak periods usually a substantial number of agents needs to be assigned to incoming calls. Consequently, the productivity of the agents, i.e., the fraction of time that agents are busy, is low during other periods. On the other hand, assigning fewer agents to incoming calls increases the productivity, but leads to longer waiting times. Hence, there is a need to balance productivity and waiting times. The solution is *call blending*, dynamically assigning agents either to incoming or outgoing traffic.

The model that we study in this note represents exactly this situation: there are s agents, the time constrained incoming calls are modeled as type 1 customers, and the type 2 calls represent the backlog of outgoing calls. The objective is to obtain a simple model that provides insight into the system behavior. Moreover, the aim is to derive simple scheduling policies that can be implemented in call center software.

Manuscript received January 3, 2002; revised July 19, 2002. Recommended by Associate Editor A. Giua.

S. Bhulai is with the Bell Laboratories, Lucent Technologies, Murray Hill, NJ 07974-0636 USA, and also with the Faculty of Sciences, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands (e-mail: sbhulai@cs.vu.nl).

G. Koole is with the Faculty of Sciences, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands (e-mail: koole@cs.vu.nl).

Digital Object Identifier 10.1109/TAC.2003.815038

The contribution of this note is twofold. We propose scheduling policies that keep part of the service capacity free for arriving time-constrained type 1 jobs. This policy mixes the traffic from the two channels such that both the waiting time constraint for type 1 jobs is met, and the throughput of type 2 jobs is maximized. In contrast to many other queueing models where idling is not optimal (see [9] for a recent survey), we show that this policy is optimal for equal service time distributions and a very good approximation otherwise.

The second contribution follows from the practical relevance of call blending in call centers. Call blending can significantly improve call center performance in many companies, compared to the traditional separation of employees in groups assigned to either incoming or outgoing traffic. However, call blending has not received much attention yet (see [6] for a recent survey on queueing models for call centers). In this note, we present a mathematical model for call blending and solve it. As the resulting policy is easily implemented, it has the potential to be used in workforce management software for call centers.

The organization of this note is as follows. In Section II, we give the exact model formulation. In Section III, we analyze the case of equal service requirements. In Section IV, we analyze the case of different service requirements.

II. Model and First Results

The exact model formulation is as follows. There are two types of traffic, type 1 and type 2, having independent exponentially distributed service requirements with rates μ_1 and μ_2. Type 1 jobs arrive according to a Poisson process with rate λ, and there is an infinite waiting queue for jobs that cannot be served yet. There is an infinite supply of type 2 jobs. There are a total of s identical servers. The long-term average waiting time of the type 1 jobs should be below a constant α. Waiting excludes the service time; if the response time is to be considered, then the average service time, $1/\mu_1$, should be added to the average waiting time. The objective for type 2 jobs is to maximize its throughput, i.e., to serve on average per unit of time as many type 2 jobs as possible, of course at the same time obeying the constraint on the type 1 waiting time.

The following control actions are possible. The moment a server finishes service, or, more generally at any moment that a server is idle, it can take one of the following three actions: start serving a type 1 job (if one or more are waiting in the queue for service), start serving a type 2 job, or remain idle.

Note that in our model preemption of jobs in service is not allowed. When preemption is allowed the problem is trivial. The optimal policy will assign all servers to type 2 jobs when no type 1 jobs are present in the system. When a type 1 job arrives then it is clearly optimal to interrupt the service of a type 2 job and to serve the type 1 job. Hence, the waiting time constraint is satisfied and the type 2 throughput is equal to $\mu_2\ (s - \lambda/\mu_1)$. Note that any work-conserving policy that satisfies the waiting time constraint is optimal and achieves the same throughput. In practice the jobs that can be preempted in call centers are e-mail messages. Therefore, it is beneficial for call centers to encourage customers to send their requests by e-mail.

This finishes the description of our model. In the next two sections, we will deal with the case $\mu_1 = \mu_2$ and $\mu_1 \neq \mu_2$, respectively. A first question that has to be answered is whether it is at all possible to find a policy that satisfies the waiting time constraint of the type 1 traffic.

Lemma 1: Let $\rho = \lambda/\mu_1$, and W^q denote the stationary waiting time of type 1 jobs. The constraint $\mathbb{E}W^q \leq \alpha$ is satisfiable when λ, μ_1, and s are such that

$$\mathbb{E}W^q = \frac{\rho^s}{\mu_1(s-1)!(s-\rho)^2} \times \left[\sum_{x=0}^{s-1} \frac{\rho^x}{x!} + \frac{\rho^s}{(s-1)!(s-\rho)}\right]^{-1} \leq \alpha.$$

Proof: It is clear that the waiting time for type 1 jobs is minimized if we schedule no type 2 jobs at all, thus reducing the model to a standard M/M/s queue. Note that the stability condition $\lambda/\mu_1 < s$ is not sufficient. Hence, $\mathbb{E}W^q$ should be calculated first, which is given by [3, Ex. 4.27] yielding the expression in the lemma. ∎

From now on, we assume that λ, μ_1, and s are such that $\mathbb{E}W^q \leq \alpha$. Checking whether this is the case can be easily done using the aforementioned formulas.

While formulating the model we stated that an idle server can schedule a type 2 or a type 1 job (when available) at any moment. Due to the fact that we are considering long-term average performance it is only optimal to schedule jobs at completion or arrival instants. Indeed, if it is optimal to keep a server idle at a certain instant, then this remains optimal until the next event in the system. This follows directly from the continuous-time Bellman equation (see [8, Ch. 11]). Therefore, it suffices to consider the system only at completion or arrival instants. Because of this, and because of the fact that the maximum total rate is uniformly bounded by $\lambda + s\max\{\mu_1, \mu_2\}$, we can use the well-known uniformization technique (see [8, Sec. 11.5]). This allows us to use discrete-time dynamic programming to compute performance measures and to find the optimal policy.

However, our system is not a standard Markov decision process (MDP), because of the different objectives for queue 1 and queue 2. The form of the problem makes it a *constrained MDP*; maximize the type 2 throughput with a constraint on the type 1 waiting time. Constrained MDPs can be solved using various techniques. Here, we use one that introduces the constraint in the objective using a Lagrange multiplier. Under weak conditions, it can be seen that the optimal policy for a certain Lagrange multiplier is optimal for the constrained problem if the value of the constraint under this policy attains exactly α. From the theory on constrained MDPs it follows that this policy is stationary and randomizes in at most 1 state. For this and other results on constrained MDPs, see [1].

III. Equal Service Requirements

Let $\mu := \mu_1 = \mu_2$. Consider the event that a server becomes idle, and that there are one or more type 1 jobs waiting. Then the controller has to choose between scheduling a type 1 or a type 2 job (or idling, but this is evidently suboptimal). Giving priority to a type 2 job and delaying type 1 jobs obviously leads to higher waiting times. Delaying the processing of a type 2 job does not change the performance for this class, as we are interested in the *long-term* throughput. This intuitive argument implies that, when a server becomes idle and a type 1 job is waiting, it is optimal to assign this type 1 job to the server. The following coupling argument shows that this is indeed true.

Theorem 2: Suppose that a server becomes idle while there are type 1 jobs waiting in the queue. Then, the action that schedules a type 1 job is among the set of optimal actions.

Proof: Let π be an arbitrary policy which respects the waiting time constraint on the type 1 jobs. Suppose that there is a time instant, say t_1, where a type 2 job is scheduled, given that there is a type 1 job waiting in the queue. Since π respects the waiting time constraint on type 1 jobs there will be a later time instant, say t_2 where this type 1 job will be scheduled.

Now, consider the policy π' which follows all actions of π except that it schedules a type 1 job at t_1 and a type 2 job at t_2. Note that this interchange does not change the decision epochs for π', since the service requirement is μ for both job types. The total number of type 2 customers served after t_2 is equal under both polices, thus also the throughput. However, the average waiting time under policy π' is the same or lower than under π, since the type 1 job is served earlier. Hence, the result follows. ∎

We model the system as a (constrained) Markov decision process. This consists of a description of the state space, the possible actions, the transition probabilities, and the reward structure. Since both types of jobs have equal service requirements, we need not distinguish between type 1 or type 2 jobs in service. Therefore, the state of the system is completely described by the number of jobs in service plus the number of type 1 jobs in the queue. Thus, the state–space is $\mathcal{X} = \mathbb{N}_0$. By Theorem 2, there cannot be less than s jobs in service while there are type 1 jobs waiting in the queue. Hence, one can only take an action in state $x \in \mathcal{X}$ if $x < s$; otherwise a type 1 job is automatically scheduled. The possible actions are denoted by $a = 0, \ldots, s - x$, corresponding to scheduling a type 2 jobs.

We denote the transition rate of going from x to y (before taking any action) by $p(x, y)$. Then we have $p(x, x-1) = \min\{x, s\}\mu$ and $p(x, x+1) = \lambda$. After such an event an action can be chosen, if the new state is below s. If action a is chosen in x (with $a \leq s - x$), then the system moves to $x + a$. Next, we uniformize the system (see [8, Sec. 11.5]). For simplicity, we assume that $s\mu + \lambda \leq 1$. (We can always get this by scaling.) Uniformizing is equivalent to adding dummy transitions (from a state to itself) such that the rate out of each state is equal to 1; then we can consider the rates to be transition probabilities.

The objectives are modeled as follows. If action a is chosen, then a reward of a is received, 1 for each type 2 job that enters service; this models the throughput. Due to the Poisson arrivals and uniformization, the average waiting time is obtained by taking the cost rates equal to the expected waiting time of an arriving customer. Thus, to obtain the average waiting costs $\mathbb{E}W^q$, we can take the cost rates equal to $[x - s + 1]^+/s\mu$, where $[x]^+ = \max\{0, x\}$. Note that the cost rates in this case are equivalent to lump costs at each epoch. The two objectives are merged into a single reward function using a Lagrange parameter $\gamma \in \mathbb{R}$ (see [1]).

In order to study structural properties of optimal policies, we define the dynamic programming operator T for functions $f\colon \mathcal{X} \to \mathbb{R}$ as follows:

$$Tf(x) = \frac{\gamma[x - s + 1]^+}{s\mu} + \sum_{y \in \{x-1, x, x+1\}} p(x, y) \max_{a \in \{0\} \cup \{1, \ldots, s-y\}} \{a + f(y + a)\}$$

with the convention that $\{1, \ldots, s - y\} = \emptyset$ if $s - y \leq 0$. Observe how the waiting time constraint is merged with the throughput by the Lagrange parameter γ. Also, note the place of the maximization: here the action depends on the state changes that occurred. (The dynamic programming operator can easily be rewritten in the standard formulation with a single overall maximization; see [4, Ch. 5]), but this would considerably complicate the notation).

The long-term average optimal actions are a solution of the optimality equation (in vector notation) $h + g = Th$. Another way of obtaining them is through value iteration, by recursively defining $V_{n+1} = TV_n$, for arbitrary V_0. For $n \to \infty$, the maximizing action converges to the optimal ones. (For existence and convergence of solutions and optimal policies, we refer to [8]).

We will next show that the optimal policy is of threshold type.

Theorem 3: There is a level c, called the threshold, such that if $x < c$, then the optimal action is $c - x$. If $x \geq c$, then 0 is the optimal action.

Proof: We first derive certain monotonicity properties of the value function V_n with $V_0 = 0$. This is easier if we rewrite T as follows:

$$Tf(x) = \frac{\gamma[x-s+1]^+}{s\mu} + \sum_{y\in\{x-1,x,x+1\}} p(x,y)U^{(s)}f(y)$$

with $U^{(s)}$ the s-fold convolution of the operator U, and U defined as

$$Uf(x) = \begin{cases} \max\{f(x), 1+f(x+1)\}, & \text{if } x < s \\ f(x), & \text{otherwise.} \end{cases}$$

It is obvious that both forms are equivalent: scheduling a group of type 2 jobs at the same time is equivalent to scheduling them one by one.

We continue with the fact that we are only interested in $\gamma \leq 0$. For $\gamma = 0$ the situation is as if the waiting time constraint poses no real constraint; for $\gamma < 0$ it pays to have a policy that tries to decrease the type 1 waiting time. For $\gamma \leq 0$, the direct "rewards" $\gamma[x-s+1]^+/s\mu$ are decreasing concave (dcv) as a function of x. The function $V_0 = 0$ is also dcv. It is straightforward to show that V_n is dcv for all n (this is equivalent to the results derived in [5, Sec. 3]). When applying U to f, we see that it is optimal to schedule a type 2 job if and only if $f(x) - f(x+1) \leq 1$. If f is dcv, then $f(x) - f(x+1)$ increases in x. Thus for each f there is a threshold level c at and above which it is not optimal to schedule type 2 jobs, and below which it is. Because the U operator is repeated s times after each event the threshold level will be reached. (In fact, there will never be scheduled more than one type 2 job, as the jump size is maximal 1). ■

We need to stress that it can be the case that it is both optimal to schedule a type 2 job and not to schedule one. This occurs if $f(x) = 1 + f(x+1)$. In this case, two threshold policies, with threshold level c and $c+1$, are both optimal, and all the policies that randomize between these two policies. We will see that, in general, to find a threshold policy that satisfies $\mathbb{E}W^q = \alpha$, we need to randomize. Next, we calculate for a fixed threshold policy that randomizes between c and $c+1$ its stationary type 1 waiting time $\mathbb{E}W^q$ and its type 2 throughput.

We assume that the policy is such that if a transition from $c+1$ to c occurs then with probability $1-\delta$ a type 2 job is immediately scheduled. This results in a transition rate $p(c+1,c)$ from $c+1$ to c of $p(c+1,c) = \delta(c+1)\mu$. The lowest possible state is c, as the state moves immediately up to c as soon as $c-1$ is reached. The other positive transition rates are $p(x,x+1) = \lambda$ for all $x \geq c$ and $p(x,x-1) = \min\{s,x\}\mu$ for all $x > c+1$. This results in a birth–death process from which the average waiting time and the throughput can be computed as follows.

Theorem 4: Let $\rho = \lambda/\mu$. Then the average waiting time as a function of the threshold c and the randomization parameter δ is given by

$$\mathbb{E}W^q_{(c,\delta)} = \frac{\rho^{s-c}c!}{\mu\delta(s-1)!(s-\rho)^2}q_c \quad \text{with } q_c$$
$$= \left[1 + \sum_{x=c+1}^{s-1}\frac{\rho^{x-c}c!}{\delta x!} + \frac{\rho^{s-c}c!}{\delta(s-1)!(s-\rho)}\right]^{-1}.$$

The throughput ξ of type 2 jobs is given by

$$\xi_{(c,\delta)} = \mu q_c\left[\sum_{x=\max\{c,1\}}^{s-1}\frac{\rho^{x-c}c!}{\delta(x-1)!} + \frac{s\rho^{s-c}c!}{\delta(s-1)!(s-\rho)}\right] - \lambda.$$

Proof: Let us calculate the stationary probabilities, which are denoted by q_x, for the birth–death process. It is readily seen that (with $\rho = \lambda/\mu$)

$$q_x = \frac{\rho^{x-c}c!}{\delta x!}q_c \text{ for all } c < x \leq s$$

and

$$q_x = \frac{\rho^{x-c}c!}{\delta s! s^{x-s}}q_c \text{ for all } x > s$$

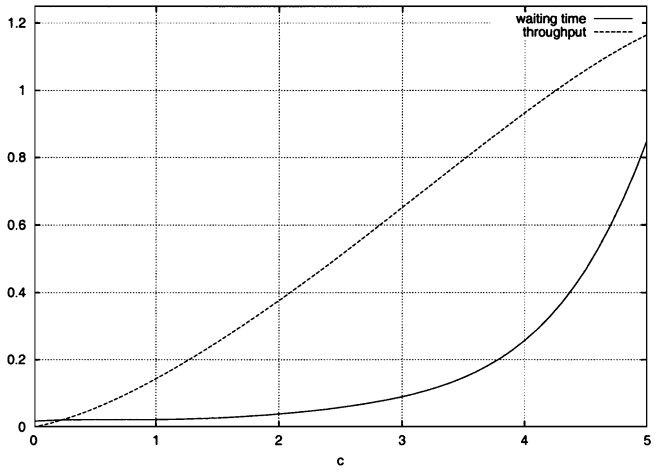

Fig. 1. $\boldsymbol{\lambda = 1/2, \mu = 1/3}$, and $\boldsymbol{s = 5}$.

with

$$q_c = \left[1 + \sum_{x=c+1}^{s-1}\frac{\rho^{x-c}c!}{\delta x!} + \frac{\rho^{s-c}c!}{\delta(s-1)!(s-\rho)}\right]^{-1}.$$

We define the probability of delay for this (c,δ) policy by

$$C_{(c,\delta)}(s,\rho) = \sum_{x=s}^{\infty} q_x = \frac{\rho^{s-c}c!}{\delta(s-1)!(s-\rho)}q_c.$$

The waiting time under the threshold policy is also completely equivalent to the one without type 2 jobs

$$\mathbb{E}W^q_{(c,\delta)} = \frac{C_{(c,\delta)}(s,\rho)}{\mu(s-\rho)}.$$

This is the formula for the type 1 waiting times. Next we derive an expression for the type 2 throughput, which we will denote by ξ. The throughput of type 2 is the total throughput minus the type 1 throughput. Therefore

$$\xi_{(c,\delta)} = \sum_{x=c}^{\infty}\mu\min\{x,s\}q_x - \lambda$$
$$= \mu q_c\left[\sum_{x=\max\{c,1\}}^{s-1}\frac{\rho^{x-c}c!}{\delta(x-1)!} + \frac{s\rho^{s-c}c!}{\delta(s-1)!(s-\rho)}\right] - \lambda.$$

The expression $\max\{c,1\}$ in the summation limit is there to prevent $(-1)!$ from appearing in case $c = 0$. ■

Fig. 1 displays the behavior of the waiting time of type 1 jobs (line) and the throughput of type 2 jobs (dotted line) when c varies. The used values of the parameters are: $\lambda = 1/2, \mu = 1/3$, and $s = 5$. The figure should be interpreted as follows: to a given waiting time guarantee α to type 1 jobs, one can read the optimal threshold value c in the figure. Next, one can read the throughput of type 2 jobs associated with α under the policy using the threshold c.

It is interesting to note that the average waiting time increases slowly, while the throughput increases nearly linearly. In the call center framework, one could linearly increase the productivity of the agents, while very little additional waiting occurs. Numerical experiments indicate that the threshold policy performs very well in case of nonexponential service times. Hence, it seems that the threshold policy is very robust.

IV. Unequal Service Requirements

When $\mu_1 \neq \mu_2$ the analysis is more complicated. In this case we have to differentiate between type 1 customers and type 2 customers.

The optimal policy will depend on these different classes and can be very complicated. From a practical viewpoint, these policies can also be difficult to implement in call center software. Therefore, we prefer to study simpler policies and we restrict ourselves to the class of threshold policies. This choice is partially supported by the intuitive reasoning behind Theorem 2. Numerical experiments indicate that this theorem also holds in case of unequal service requirements. The restriction to the class of threshold policies forces the policies to be simple and appealing. Moreover, we will show by numerical computation that threshold policies are a good approximation to the optimal policy.

Let x denote the number of type 1 jobs in service and in the queue, and let y denote the number of type 2 jobs in service. Let the threshold c be fixed, and let us for the moment assume that we do not randomize in states $x + y = c + 1$. Then the stationary probabilities $q_{x,y}$ are determined by the following set of equilibrium equations.

For $y = c$ we have

$$((s-c)\mu_1 + \lambda + c\mu_2)q_{x,c} = \lambda q_{x-1,c} + (s-c)\mu_1 q_{x+1,c} \quad x \geq s-c \tag{1}$$

$$(x\mu_1 + \lambda + c\mu_2)q_{x,c} = \lambda q_{x-1,c} + (x+1)\mu_1 q_{x+1,c} \quad 0 < x < s-c \tag{2}$$

$$\lambda q_{0,c} = \mu_1 q_{1,c} + \mu_1 q_{1,c-1}. \tag{3}$$

For $0 < y < c$, the set of equations becomes

$$\begin{aligned}((s-y)\mu_1 + \lambda + y\mu_2)q_{x,y} &= \lambda q_{x-1,y} + (s-y)\mu_1 q_{x+1,y} \\ &\quad + (y+1)\mu_2 q_{x,y+1} \quad x \geq s-y \end{aligned} \tag{4}$$

$$\begin{aligned}(x\mu_1 + \lambda + y\mu_2)q_{x,y} &= \lambda q_{x-1,y} + (x+1)\mu_1 q_{x+1,y} \\ &\quad + (y+1)\mu_2 q_{x,y+1} \quad c-y < x < s-y \end{aligned} \tag{5}$$

$$\begin{aligned}((c-y)\mu_1 + \lambda)q_{c-y,y} &= (y+1)\mu_2 q_{c-y,y+1} + (c-y \\ &\quad + 1)\mu_1 q_{c-y+1,y} \\ &\quad + (c-y+1)\mu_1 q_{c-y+1,y-1}. \end{aligned} \tag{6}$$

Finally, for $y = 0$, we have

$$(s\mu_1 + \lambda)q_{x,0} = \lambda q_{x-1,0} + s\mu_1 q_{x+1,0} + \mu_2 q_{x,1} \quad x \geq s \tag{7}$$

$$(x\mu_1 + \lambda)q_{x,0} = \lambda q_{x-1,0} + (x+1)\mu_1 q_{x+1,0} + \mu_2 q_{x,1} \quad c < x < s \tag{8}$$

$$(c\mu_1 + \lambda)q_{c,0} = \mu_2 q_{c,1} + (c+1)\mu_1 q_{c+1,0}. \tag{9}$$

This infinite set of equilibrium equations can be numerically solved by using the matrix-geometric approach developed by Neuts [7] or by the spectral expansion method as described by Chakka and Mitrani [2]. However, note that in the equations there is no flow from $q_{x,y}$ toward level $y+1$ or higher when $x > c-y$, since jobs of type 1 are given priority over type 2 jobs. Due to this special structure of the equations, we can solve this part of the system analytically using standard results from the theory of linear difference equations. The equilibrium equations are solved by solving the equations for $y = c$ and afterwards working the way down from $y = c-1$ to $y = 0$. After solving these equations, we obtain a finite set of equations still to be solved. However, these equations can easily be computed numerically by applying the recurrence relations on the obtained solutions.

Theorem 5: The solution to (1), (4), and (7) is given by

$$q_{x,y} = \sum_{i=y}^{c} K_{y,i-y} z_i^{x-(s-i)} \quad 0 \leq y \leq c, \qquad x \geq s-y. \tag{10}$$

The constants z_i for $i = 0, \ldots, c$ are the solution to a homogeneous difference equation and are given by the first equation shown at the bottom of the page. The constants $K_{y,i}$ are given by the recurrence relation

$$K_{y,i} = \frac{(y+1)\mu_2}{i(\mu_1+\mu_2) - i\mu_1 z_{y+i}} K_{y+1,i-1} \quad 0 \leq y < c, \qquad 1 \leq i \leq c-y. \tag{11}$$

Proof: Focus the attention on a particular row y with $0 \leq y \leq c$. Consider the corresponding homogeneous equations associated with y and $x > s-y$; thus, we are considering the homogeneous parts of (1), (4), or (7). Its solution is given by $q_{x,y} = q_{s-y,y} z_y^{x-(s-y)}$, where z_y is the root of the polynomial

$$\lambda - ((s-y)\mu_1 + \lambda + y\mu_2)z + (s-y)\mu_1 z^2.$$

By straightforward computation, one shows that one of the two roots is larger than 1 and, therefore, not useful. The other root, which is positive and less than one, is given by the second equation shown at the bottom of the page. Note that the $c+1$ roots $z_0, \ldots, z_c$ are different from each other.

Now, we will show that (10) and (11) satisfy the difference equations. The case $y = c$ has already been considered during the derivation of z_c, since (1) is a homogeneous difference equation. Now, let $0 \leq y < c$ and consider (4) and (7) given by

$$\lambda q_{x-1,y} - ((s-y)\mu_1 + \lambda + y\mu_2)q_{x,y} + (s-y)\mu_1 q_{x+1,y} + (y+1)\mu_2 q_{x,y+1}.$$

Substitute expression (10) into this equation. Fix $1 \leq i \leq c-y$ and look at all factors z_{y+i}. This yields the following expression

$$K_{y,i} z_{y+i}^{x-(s-y-i)-1} \left[\lambda - ((s-y)\mu_1 + \lambda + y\mu_2)z_{y+i} + (s-y)\mu_1 z_{y+i}^2\right] + (y+1)\mu_2 K_{y+1,i-1} z_{y+i}^{x-(s-y-i)}.$$

Now subtract the following quantity from this:

$$K_{y,i} z_{y+i}^{x-(s-y-i)-1} \left[\lambda - ((s-y-i)\mu_1 + \lambda + (y+i)\mu_2)z_{y+i} + (s-y-i)\mu_1 z_{y+i}^2\right].$$

$$z_i = \frac{((s-i)\mu_1 + \lambda + i\mu_2) - \sqrt{((s-i)\mu_1 + \lambda + i\mu_2)^2 - 4\lambda(s-i)\mu_1}}{2(s-i)\mu_1}.$$

$$z_y = \frac{((s-y)\mu_1 + \lambda + y\mu_2) - \sqrt{((s-y)\mu_1 + \lambda + y\mu_2)^2 - 4\lambda(s-y)\mu_1}}{2(s-y)\mu_1}.$$

This quantity is equal to zero, since the terms between the brackets is the solution to the homogeneous difference equation for level $y+i$. The result of the subtraction is given by

$$K_{y,i} z_{y+i}^{x-(s-y-i)-1} \left[-i(\mu_1+\mu_2) z_{y+i} + i\mu_1 z_{y+i}^2\right] + (y+1)\mu_2 K_{y+1,i-1} z_{y+i}^{x-(s-y-i)}.$$

which is equal to

$$z_{y+i}^{x-(s-y-i)}[-(i(\mu_1+\mu_2) - i\mu_1 z_{y+i})K_{y,i} + (y+1)\mu_2 K_{y+1,i-1}].$$

Substituting (11) into this equation yields zero, showing that (10) and (11) indeed are the solution to (1), (4), and (7). ■

At this point (1), (4), and (7) have been solved up to $K_{0,0}, \ldots, K_{c,0}$. We still have a finite number of equations to solve, which can be done numerically as follows. Using relations (2), (5), and (8), one can compute the values of $q_{x,y}$ for $0 \le y \le c$ with $c - y < x < s - y$ expressed in the unknown constants. Then the boundary conditions (3), (6), and (9) relate the constants to each other leaving only one constant to be determined. This constant is finally determined by the condition $\sum_{x,y} q_{x,y} = 1$. Now the complete system is determined.

Once the probabilities $q_{x,y}$ are known, the performance characteristics of Section III for the two types of jobs can be easily computed. The probability of delay is given by

$$C_{(c)}(s) = \sum_{y=0}^{c} \sum_{x=s-y}^{\infty} q_{x,y} = \sum_{y=0}^{c} \sum_{i=y}^{c} \frac{K_{y,i-y}}{1-z_i} z_i^{i-y}.$$

Recall that $[x]^+ = \max\{0, x\}$, and define $W(x,y)$ recursively by

$$W(x,y) = \frac{1}{(s-y)\mu_1 + y\mu_2}[1 + (s-y)\mu_1 W(x-1,y) + y\mu_2 W(x,[y-1]^+)]$$

for $x+y \ge s$, and 0 otherwise. Note that $W(x,y) = (x+y-s+1)^+/s\mu$ when $\mu = \mu_1 = \mu_2$. The waiting time of type 1 jobs is given by

$$\mathbb{E}W_{(c)}^q = \sum_{x,y} W(x,y) q_{x,y}.$$

Finally, the throughput of type 2 jobs is given by

$$\xi_{(c)} = \mu_2 \sum_{x,y} y q_{x,y}.$$

Note that the previous results are for the case with no randomization. When randomizing in states $x+y = c+1$, the previous results still hold with some minor changes. Randomization alters the balance equations for the states $x+y = c+1$ and $x+y = c$. Since this does not affect (1), (4), and (7), Theorem 5 is still valid where the constants K now depend on δ, the randomization probability. The finite set of equations can now be solved recursively as described before. Hence, the waiting time of type 1 jobs and the throughput of type 2 jobs will also depend on δ.

Fig. 2 illustrates the behavior of the waiting time of type 1 jobs (line) and the throughput of type 2 jobs (dotted line) when $\mu_1 \neq \mu_2$. The chosen parameters in this case are $\lambda = 1/2, \mu_1 = 4/10, \mu_2 = 3/10$, and $s = 5$.

Again, we see that the average waiting time increases slowly, while the throughput increases nearly linearly. Since the threshold policy does not need to be the optimal policy, it is interesting to numerically compute the performance of the optimal policy. This can be done by using the dynamic programming operator for a fixed value of the Lagrange parameter γ yielding a policy π. The waiting time of type 1 jobs is obtained by iterating the dynamic programming operator following policy π without the reward rate for scheduling type 2 jobs. Similarly, the throughput of type 2 jobs is obtained by iterating the dynamic programming operator following policy π without the cost rates for waiting. For the fixed value of the Lagrange parameter γ, we thus obtain the optimal waiting time with the corresponding throughput (for results on optimality, see [1]).

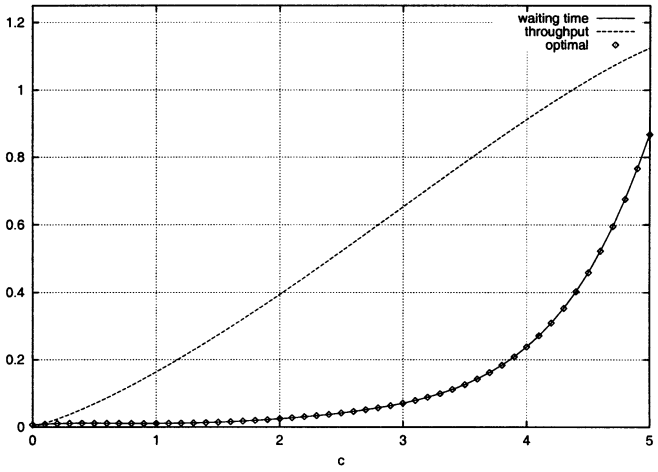

Fig. 2. $\lambda = 1/2, \mu_1 = 4/10, \mu_2 = 3/10$, and $s = 5$.

The graph of the optimal policy is generated by computing the waiting times with the corresponding throughput for various values of the Lagrange parameter γ. Matching the throughput with the throughput computed by the threshold policy yields a value of c. The minimal waiting time that can be achieved for that level c is the waiting time computed by the optimal policy. This waiting time is denoted by a dot in the graph.

It is interesting to note that the optimal policy yields a performance very close to the approximative threshold policy. Extensive experiments show that for other parameter values the same result is obtained. The experiments indicate that the optimal policy behaves nearly as a threshold policy, but minor differences occur when $x+y$ is close to the threshold value.

REFERENCES

[1] E. Altman, *Constrained Markov Decision Processes*. London, U.K.: Chapman and Hall, 1999.
[2] R. Chakka and I. Mitrani, "Heterogeneous multiprocessor systems with breakdowns: Performance and optimal repair strategies," *Theoretical Comput. Sci.*, vol. 125, pp. 91–109, 1994.
[3] R. B. Cooper, *Introduction to Queueing Theory*. Amsterdam, The Netherlands: North-Holland, 1981.
[4] G. M. Koole, *Stochastic Scheduling and Dynamic Programming*. Amsterdam, The Netherlands: CWI Tract 113. CWI, 1995.
[5] ——, "Structural results for the control of queueing systems using event-based dynamic programming," *Queueing Syst.*, vol. 30, pp. 323–339, 1998.
[6] G. M. Koole and A. Mandelbaum, "Queueing models of call centers: An introduction," *Ann. Oper. Res.*, vol. 112, 2002, to be published.
[7] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models*. Baltimore, MD: Johns Hopkins Univ. Press, 1981.
[8] M. L. Puterman, *Markov Decision Processes*. New York: Wiley, 1994.
[9] S. Stidham Jr., "Analysis, design, and control of queueing systems," *Oper. Res.*, vol. 50, pp. 197–216, 2002.