Scheduling Time-Constrained Jobs in the Presence of Background Traffic

Sandjai Bhulai and Ger Koole
Vrije Universiteit
Department of Mathematics and Computer Science
De Boelelaan 1081a, 1081 HV Amsterdam
The Netherlands
E-mail: {sbhulai, koole}@cs.vu.nl

Abstract

In this paper we study the scheduling of jobs with a constraint on the average waiting time in the presence of background jobs. The objective is to schedule to s servers such that the throughput of the background traffic is maximized while satisfying the response time constraint on the foreground traffic.

The arrivals are determined by a Poisson process and the service times of the jobs are independent exponentially distributed. We consider both the situation where service requirements by both types of jobs are equal and unequal. The first situation is solved to optimality, for the second situation we find the best policy within a certain class of policies. Optimal schedules always keep part of the service capacity free for arriving foreground jobs. Applications of this model can be found in computer systems, communication networks and call centers.

1 Introduction

In this paper 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. In addition 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 non-preemptive regime. The question that we will answer in this paper 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 free and there are no type 1 jobs in the queue, whether this server should start on 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.

The organization of this paper is as follows. In the next section we give several situations where our model could be useful. Then we give the exact model formulation. In Section 4 we analyze the case of equal service requirements, in Section 5 we analyze the case with different service requirements.

2 Motivation

The model that we study in this paper can be used to analyze many different real-life systems. We will discuss three of them.

Scheduling in parallel computer systems

Consider a parallel computer system with s single-tasking processing units, in which there are time-constrained jobs and background jobs. The time-constrained jobs are for example those tasks for which the user is waiting, thus the response time, and therefore the waiting time should be below a certain bound. The background jobs can be management jobs that should be run regularly, but without degrading the response times of the user jobs.

An example is the following. The Java runtime environment deletes objects when it determines that they are no longer being used. This process is called "garbage collection". The more often garbage is collected, the better it is, but only without deteriorating the response times of the user processes. Thus processes that execute the garbage collection can well be modeled as type 2 traffic, while the user processes are modeled as type 1 traffic.

Admission control in communication networks

In many voice or data transmission systems the available bandwidth is divided into "channels". Examples are time-domain multiplexing or frequency-domain multiplexing. In such cases we can think of a communication line as divided into s parallel channels. But also in ATM such a situation can occur. Suppose that many low bandwidth connections are multiplexed into one high bandwidth connection. Then the capacity needed on the outgoing link for traffic coming in over one of the incoming lines need not exceed its bandwidth, it suffices to reserve bandwidth equal to the capacity

of the incoming lines. This motivates the division of the service capacity in several parallel servers.

Examples of different types of traffic can easily be found in communication systems. Voice or web pages have constraints on the response times. For example for E-mail there are no such constraints, and the throughput is its major performance measure. This gives a natural division in type 1 and type 2 traffic. In the context of ATM networks this is even formalized, with the existence of CBR/VBR traffic with constraints on the response times, and ABR traffic without.

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 designate 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 amount of agents needs to be assigned to incoming calls. This leads to a low productivity of the agents during other periods. On the other hand, designating less 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 paper 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.

3 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 in total s 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, μ_1^{-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 a server is free, 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;
- remain idle.

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; but first make several general observations considering this problem.

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. It is clear that the waiting time for type 1 jobs is minimized if we schedule no type 2 jobs at all. Without the type 2 jobs the system becomes a standard M/M/s queue. If we denote with W^q the stationary waiting time of this queue, then the question is whether $\mathbb{E}W^q \leq \alpha$. If the system is unstable, i.e., $\lambda/\mu_1 \geq s$, then $\mathbb{E}W^q = \infty$. But the condition $\lambda/\mu_1 < s$ is not sufficient. Thus $\mathbb{E}W^q$ should first be calculated. Define $C(s,\rho)$ as the stationary delay probability, for s servers and a load of $\rho = \lambda/\mu_1$ Erlang. Then $\mathbb{E}W^q$ is given by the well-known formula

$$\mathbb{E}W^q = \frac{C(s,\rho)}{\mu_1(s-\rho)},$$

with $C(s,\rho)$ given by

$$C(s,\rho) = \frac{\rho^s}{(s-1)!\,(s-\rho)} \left[\sum_{x=0}^{s-1} \frac{\rho^x}{x!} + \frac{\rho^s}{(s-1)!\,(s-\rho)} \right]^{-1}. \quad (1)$$

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 easily be done using the formulas above.

We go back to the original problem with jobs of type 2. Consider that a server becomes free, 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 the 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 free and a type 1 job is waiting, it is optimal to assign this type 1 job to the server. For $\mu_1 = \mu_2$ this statement can be proven with a simple coupling argument. The situation for $\mu_1 \neq \mu_2$ is more complicated; still we only consider policies that give priority to type 1 jobs. The reason is the intuitive argument above and the fact that it forces policies to be simple and appealing.

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 $\mu_2(s-\lambda_1/\mu_1)$. Note that any work-conserving policy that satisfies the waiting time constraint is optimal and achieves the same throughput.

While formulating the model we stated that a free 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. 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. 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 MDP's 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 MDP's it follows that this policy is stationary and randomizes in at most 1 state. For this and other results on constrained MDP's, see the recent book of Altman [1].

4 Equal service requirements

If $\mu_1 = \mu_2 =: \mu$, then we do not have to 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 $\mathcal{X} = \mathbb{N}_0$. There is only a choice in state $x \in \mathcal{X}$ if $x < s$; otherwise a type 1 job is automatically scheduled. The possible actions are denoted with $a = 0, \ldots, s-x$, corresponding to scheduling a type 2 jobs. We denote the transition rate of going from x to y (before taking any action) with $p_t(x,y)$. Then we have $p_t(x,x-1) = \min\{x,s\}\mu$ and $p_t(x,x+1) = \lambda$. After such an event an action can be chosen, if the new state is below s. If action a is chosen in x (with $a \le s-x$), then the system moves to $x+a$.

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. By Little's law, the average waiting time is related to the average number of jobs in queue L^q, by the following relation: $\mathbb{E}L^q = \lambda\,\mathbb{E}W^q$. Thus to obtain the average waiting costs $\mathbb{E}W^q$ we can take rate costs equal to $(x-s)^+/\lambda$.

Next we uniformize the system. For simplicity we assume that $s\mu + \lambda \le 1$. (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. Note also that rate costs 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 γ. Define the dynamic programming operator T as follows:

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

with the convention that $\{1,\ldots,s-y\} = \emptyset$ if $s-y \le 0$. Note the place of the maximization: here the action depends on the state changes that occurred. (The dp operator can easily be rewritten in the standard formulation with a single overall maximization (see Chapter 5 of Koole [3]), 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 Puterman [6].)

We will next show that the optimal policy has the following form. There is a level c, called the threshold, such that if $x < c$, then the optimal action is $c - x$. If $x \ge c$, then 0 is the optimal action. To show this 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:

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

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

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

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 $\gamma \le 0$. 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 $\gamma \le 0$ the direct "rewards" $\gamma(x-s)^+/\lambda$ 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 in Section 3 of Koole [4]). When applying U to f, we see that it is optimal to schedule a type 2 job if and only if $f(x) - f(x+1) \le 1$. 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 that we study next.

Let us calculate the stationary probabilities, which are denoted with q_x. 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

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

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

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

If $c = 0$ and $\delta = 1$ then no type 2 jobs are admitted, and indeed $C_{(0,1)}(s,\rho) = C(s,\rho)$ is given by (1). 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 with ξ. The throughput of type 2 is the total throughput minus the type 1 throughput. Therefore

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

Figure 1 displays the behaviour 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: given a waiting time guarantee α to type 1 jobs, the throughput c of type 2 jobs under the optimal policy can be read in the figure. It is interesting to note that the average waiting time increases slowly, while the throughput increases nearly linearly.

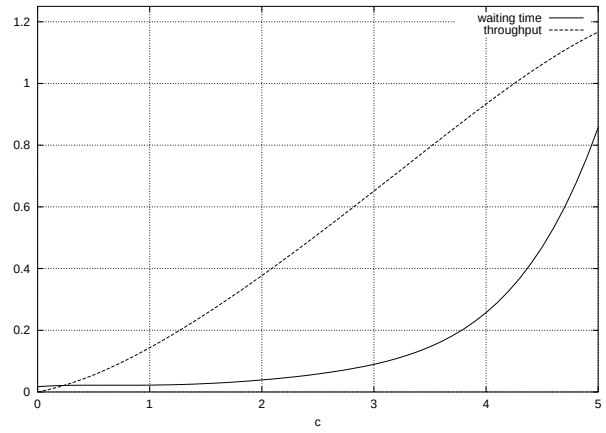

Figure 1: $\lambda = 1/2$, $\mu = 1/3$ and $s = 5$.

5 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. 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 in Section 3. 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. Then the stationary probabilities $q_{x,y}$ are determined by the following set of equilibrium equations. For $y = c$ we have

$$\begin{aligned}
\big((s-c)\mu_1 + \lambda + c\mu_2\big)\, q_{x,c} &= \lambda\, q_{x-1,c} + (s-c)\mu_1\, q_{x+1,c} && x \geq s-c, \quad (2)\\
(x\mu_1 + \lambda + c\mu_2)\, q_{x,c} &= \lambda\, q_{x-1,c} + (x+1)\mu_1\, q_{x+1,c} && 0 < x < s-c, \quad (3)\\
\lambda\, q_{0,c} &= \mu_1\, q_{1,c} + \mu_1\, q_{1,c-1}. && (4)
\end{aligned}$$

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

$$\begin{aligned}
\big((s-y)\mu_1 + \lambda + y\mu_2\big)\, q_{x,y} &= \lambda\, q_{x-1,y} + (s-y)\mu_1\, q_{x+1,y} + \\
&\quad (y+1)\mu_2\, q_{x,y+1} && x \geq s-y, \quad (5)\\
(x\mu_1 + \lambda + y\mu_2)\, q_{x,y} &= \lambda\, q_{x-1,y} + (x+1)\mu_1\, q_{x+1,y} + \\
&\quad (y+1)\mu_2\, q_{x,y+1} && c-y < x < s-y, \quad (6)\\
\big((c-y)\mu_1 + \lambda\big)\, q_{c-y,y} &= (y+1)\mu_2\, q_{c-y,y+1} + \\
&\quad (c-y+1)\mu_1\, q_{c-y+1,y} + (c-y+1)\mu_1\, q_{c-y+1,y-1}. && (7)
\end{aligned}$$

Finally for $y = 0$ we have

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

$$(x\mu_1 + \lambda)\, q_{x,0} = \lambda\, q_{x-1,0} + (x+1)\mu_1\, q_{x+1,0} + \mu_2\, q_{x,1} \qquad c < x < s, \tag{9}$$

$$(c\mu_1 + \lambda)\, q_{c,0} = \mu_2\, q_{c,1} + (c+1)\mu_1\, q_{c+1,0}. \tag{10}$$

This infinite set of equilibrium equations can be numerically solved by using the matrix-geometric approach developed by Neuts [5] or by the spectral expansion method as described by Chakka and Mitrani [2]. However, note that in the equations there is no flow from $q_{x,y}$ towards 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, one 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.

Focus the attention on a particular row 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 (2), (5) or (8). 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 $\lambda - \big((s-y)\mu_1 + \lambda + y\mu_2\big)\, z + (s-y)\mu_1\, z^2$. One can show 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

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

Note that the $c+1$ roots $z_0, \ldots, z_c$ all differ from each other. Substituting the solution of $q_{x,c}$ into (5) with $y = c-1$ results in an inhomogeneous difference equation with general solution $q_{x,c-1} = K_{c-1,0}\, z_{c-1}^{x-(s-c+1)} + K_{c-1,1}\, z_c^{x-(s-c)}$, where $K_{c-1,0}$ and $K_{c-1,1}$ are constants to be determined. The first term is the solution of the homogeneous part and the second is a particular solution of the inhomogeneous equation. Next we can substitute this solution into (5) again with $y = c-2$ and work all the way down. This leads to the general solution of equations (2), (5) and (8) which is given by

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

with

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

At this point equations (2), (5) and (8) 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 solved numerically as follows. Using relations (3), (6) and (9) one can compute the values of $q_{x,y}$ for $0 \leq y \leq c$ with $c - y < x < s - y$ expressed in the unknown constants. Then the boundary conditions (4), (7) and (10) relate the constants to each other leaving only one constant to be determined. This constant is finally determined by the condition $\sum_{x,y} q_{x,y} = 1$ and the complete system is determined.

Once the probabilities $q_{x,y}$ are known, the performance characteristics of Section 4 for the two types of jobs can be easily computed. The probability of delay is given by

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

The waiting time of type 1 jobs is given by

$$\mathbb{E}W_{(c)}^{q} = \frac{1}{\mu_1} \sum_{x,y} \big[(x+y) - s\big]^+ q_{x,y},$$

with $[x]^+ = \max\{x, 0\}$. And finally the throughput of type 2 jobs is given by

$$\xi = \mu_2 \sum_{x,y} y\, q_{x,y}.$$

Figure 2 illustrates the behaviour 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 $s = 5$.

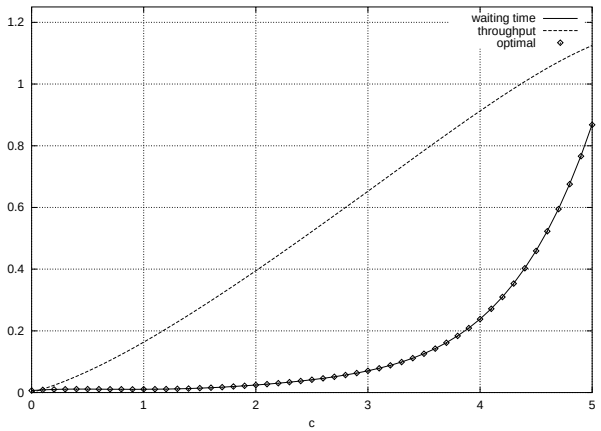

Figure 2: $\lambda = 1/2$, $\mu_1 = 4/10$, $\mu_2 = 3/10$ and $s = 5$.

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. The graph for the optimal policy is generated as follows.

Fix the value of c and compute the corresponding throughput and waiting time under the threshold policy. The optimal policy gives us pairs of throughputs and waiting times. The pair with the same throughput as under the threshold policy thus has the minimum achievable waiting time for this throughput. This waiting time is denoted with 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*. Chapman and Hall, 1999.

[2] R. Chakka and I. Mitrani. Heterogeneous multiprocessor systems with breakdowns: performance and optimal repair strategies. *Theoretical Computer Science*, 125:91–109, 1994.

[3] G.M. Koole. Stochastic scheduling and dynamic programming. CWI Tract 113. CWI, Amsterdam, 1995.

[4] G.M. Koole. Structural results for the control of queueing systems using event-based dynamic programming. *Queueing Systems*, 30:323–339, 1998.

[5] M.F. Neuts. *Matrix-geometric solutions in stochastic models*. Johns Hopkins University Press, 1981.

[6] M.L. Puterman. *Markov Decision Processes*. John Wiley & Sons, 1994.