Scheduling services in a queuing system with impatience and setup costs

Alain Jean-Marie, Emmanuel Hyon

Scheduling services in a queuing system with impatience and setup costs

Alain Jean-Marie[1] and Emmanuel Hyon[2]

[1] INRIA and LIRMM, CNRS/Université Montpellier 2, 161 Rue Ada, F-34392 Montpellier, ajm@lirmm.fr
[2] Université Paris Ouest Nanterre la Défense, LIP6, UPMC 4 place Jussieu, F-75252 Paris Cedex, Emmanuel.Hyon@u-paris10.fr.

Abstract. We consider a single server queue in discrete time, in which customers must be served before some limit sojourn time of geometrical distribution. A customer who is not served before this limit leaves the system. The fact of serving customers, holding them in queue or losing them induce costs. The purpose is to decide when to serve the customers so as to minimize these costs. We use a Markov Decision Process with infinite horizon and discounted criterion. We establish the structural properties of the stochastic dynamic programming operator, and we deduce that the optimal policy is of threshold type, and we compute the threshold explicitly.
Keywords: Scheduling, queuing system, impatience, deadline, optimal control, Markov decision processes

1 Introduction

In this paper we are interested in the optimal control of a queuing system with impatient customers (or, equivalently said, customers with deadlines). The set-up of customer services, the storage of the customers in the queue as well as their "loss" (departure from the queue due to impatience) induce some costs and it has to be decided when to begin the service in order to minimize these costs.

Controlled queuing models, deterministic as well as stochastic, have been largely studied in the literature since their application fields are numerous. Nevertheless most of these works do not consider impatient customers but rather losses due to overflow. Yet, the phenomenon of impatience, associated with deadlines or "timeouts", has become non negligible in several fields of engineering.

The literature features papers on the performance evaluation of queues with impatience, but none of them seem to address the case of choosing whether to serve or not, in the presence of setup costs. The problem of optimally controlling a batch server in a queue (without impatience) has been addressed in [1, 3] (see also the references therein, and see [2] for further references). Its resolution is based on the Markov Decision Process formalism, and goes through establishing some structural properties of the value function and the dynamic programming operator. This then allows to deduce that the solution is a threshold policy. It appears that extending the techniques developed in [3] to queues with impatience

is not straightforward, because impatience tends to destroy the structural properties that are commonly used for proving the optimality of threshold policies. In this paper, we show that structural properties exist despite the occurrence of losses and present the solution for service batches of unit size. For this purpose, we use some tools which, in our opinion, will be useful for solving more complex cases.

More precisely: we adapt the framework of structural analysis of Markov Decision Processes, as described for instance in [4]. The model and the cost structure are described in Section 2. We establish in Section 3, the structural properties of the stochastic dynamic programming operator and we show that the optimal policy is a threshold policy. Furthermore, we explicitly compute the threshold value as a function of the parameters. We discuss in our conclusion some problems encountered with general batch sizes. Details are provided in [2].

2 Model

We proceed with introducing the model, and formulating the optimal control problem in the framework of Markov Decision Processes, using the notation of Puterman [4]. Due to space limitations, some notations and concepts are quoted from this reference, to which the reader is directed for formal definitions.

2.1 System Dynamics

We consider a discrete time (or slotted) model, where the slot is the time unit. Customers arrive at the beginning of each slot. They are stored in an infinite buffer in which they wait for to be admitted in the server to be processed. This admission decision is made by a controller. The service duration is one time slot.

Denote with A_n the number of arrivals at the beginning of slot n. The sequence $\{A_n\}_{n\in\mathbb{N}}$ is assumed to be an i.i.d. sequence of random variables. With the usual abuse of notation, we denote generically this common distribution with A. We furthermore assume that A is of mean λ.

The admission decision of the controller takes place just after arriving customers have been taken into account. We call $x_n \in \mathbb{N}$ the number of waiting customers at that epoch in slot n. The set of decisions, or action space, is denoted with $\mathcal{Q} = \{0, 1\}$, where $q_n = 1$ if one customer is admitted and 0 otherwise. We assume that the controller may choose $q_n = 1$ even if $x_n = 0$, which has no effect. The number of customers remaining in the buffer just after the decision is then $y_n = (x_n - q_n)^+$, with $x^+ = \max(0, x)$.

During a slot, losses can occur because customers become impatient and leave. It is assumed that each customer in the buffer has a constant probability $\alpha \in [0, 1]$ of leaving in each slot, independently from the past and from other customers. This is equivalent to assuming that the patience of each customer is geometrically distributed on $\mathbb{N}$ with parameter α. Customers in service are not impatient. For notational convenience, we introduce the stochastic operators $I(y)$ and $S(y)$ which count, respectively, the number of customers lost (impatient) and

remaining (survivors), out of y present at the beginning of a slot. Conditioned on the value of $y_n = y$, $I(y)$ and $S(y)$ are Binomial random variables with respective means αy, and $\overline{\alpha} y$, where $\overline{\alpha} = 1 - \alpha$, and $I(y) + S(y) = y$. With this notation, the evolution of the state from slot n to slot $n+1$ is given by the recurrence equation:

$$x_{n+1} \;=\; R(x_n, q_n) \;:=\; S\left((x_n - q_n)^+\right) + A_{n+1} \;, \tag{1}$$

whereas the number of customers lost in slot n is equal to $I((x_n - q_n)^+)$.

We shall use the following property, in which $\geq_{st}$ refers to the usual (strong) stochastic ordering between random variables.

Proposition 1. *For any $x \geq y \in \mathbb{N}$ we have $S(x) \geq_{st} S(y)$. If $X \geq_{st} Y$, then $S(X) \geq_{st} S(Y)$.*

2.2 Elements of the Markov Decision Process

Transition probabilities. The dynamics of the controlled process are characterized by the probabilities to move in state z, given that the state is y and the action is $q \in \mathcal{Q}$: $\mathbb{P}\left(z|(y,q)\right) = \mathbb{P}\left(x_{n+1} = z | x_n = y, q_n = q\right)$. These probabilities do not depend on n. Their exact expression is not relevant to our analysis, which is based on the recurrence (1).

Rewards/Costs. The costs associated with decisions and transitions are the following. First, there is a setup cost c_B which is incurred when the controller chooses to admit one customer into service. Second, there is a cost associated to each customer leaving the queue due to impatience: at slot n, it is $c_L I(y_n)$, where c_L is the cost of a single loss. Finally, there is a holding cost c_H per remaining customer. We assume that it applies to all customers present after the service admission decision, so that the cost for slot n is $c_H y_n$. The total average cost incurred by taking decision q when the state is x, is then the function of (x, q):

$$c(x,q) \;=\; q\, c_B \;+\; \left(c_L\, \alpha + c_H\right)(x-q)^+ \;=\; q\, c_B \;+ c_C\,(x-q)^+ \;, \tag{2}$$

where $c_C = \alpha c_L + c_H$ is the *per-capita* cost for customers. Observe that this cost function is *not* bounded, unless $c_C = 0$.

Dynamic programming. We consider a discounted cost criterion and the discount factor is denoted by θ. We make this choice in order to avoid the complexities associated with the average cost criterion. Under each policy π, the evolution of the system generates a random sequence of states x_n and decisions q_n. The value function of policy π is then defined as:

$$v_\theta^\pi(x) \;=\; \mathbb{E}_x^\pi\left[\sum_{n=0}^{\infty} \theta^n\, c(x_n, q_n)\right] \;,$$

where $x_0 = x$. Our aim is to find the optimal policy π^* (in some adequate set of policies) such that $\forall x \in \mathbb{N}$, $v_\theta^{\pi^*}(x) = v_\theta^*(x) = \min_\pi v_\theta^\pi(x)$. This policy is provided by any solution to the *dynamic programming equation* $v_\theta = \min_q(T v_\theta)$, where the operator T, acting on functions v, is:

$$(Tv)(x,q) \;=\; c(x,q) \;+\; \theta\; \mathbb{E}\left[v\left(R(x,q)\right)\right] \;. \tag{3}$$

3 The Optimal Policy

In this part we study the structural properties of value functions in order to get qualitative results on the optimal policy. Specifically, we prove that the optimal policy is of threshold type.

The framework is that of property propagation through the Dynamic Programming operator. It consists in three steps: first identify two related sets of "structured" value functions, V^σ and policies D^σ: if the value function belongs to V^σ, then the optimal policy belongs to D^σ. Then show that the properties of v are conserved (or "propagated") by the operator T. At last, check that these properties are kept when passing to the limit. A *structure theorem* then allows to ensure that there exists an optimal policy and states, at the same time, that this optimal policy can be chosen in the set of structured policies. In the present case, the properties involved are increasingness, convexity and submodularity. The structured policies are the monotone ones. For easier reference, the methodological framework useful to our analysis is gathered in [2]. This includes the notion of *submodularity*: a real-valued function g defined on two partially ordered sets $\mathcal{X}\times\mathcal{Q}$ is called submodular if, for any $\overline{x}\geq\underline{x}\in\mathcal{X}$ and any $\overline{q}\geq\underline{q}\in\mathcal{Q}$:

$$g(\overline{x},\overline{q})-g(\underline{x},\overline{q})\leq g(\overline{x},\underline{q})-g(\underline{x},\underline{q}).$$

To submodular operators will correspond monotone decision functions.

3.1 Structural Properties of the Dynamic Programming Operator

In this part we establish structural results of the dynamic programming operator for our system: propagation of monotonicity, submodularity and convexity.

Lemma 1. *Let $\tilde{v}$ be the function defined by $\tilde{v}(x)=\min_q Tv(x,q)$ for any $x\in\mathbb{N}$. Then $\tilde{v}$ is nondecreasing in x if v is nondecreasing in x.*

Proof. The definition of Tv in (3) involves two terms given in Eqs. (2) and (1). We show first that the costs $c(x,q)$ are nondecreasing for a given decision q. Indeed, from Equation (2) the cost is either equal to $(x-1)c_C+c_B$ or xc_C which are nondecreasing in x. Then, from Proposition 1, it follows that $S\left((x+1-q)^+\right)\geq_{st}S\left((x-q)^+\right)$. Therefore we have that $R(x+1,q)\geq_{st}R(x,q)$, which implies $\mathbb{E}v(R(x+1,q))\geq\mathbb{E}v(R(x,q))$ since v is increasing. As a consequence, the function $Tv(x,q)$ is the sum of two increasing functions of x for every q. The minimum over q is therefore also increasing.

Lemma 2 (Submodularity). *For any nondecreasing convex function v, the function $Tv(x,q)$ is submodular on $\mathbb{N}\times\mathcal{Q}$.*

Proof. We shall show that $\Delta_q Tv(x):=Tv(x,1)-Tv(x,0)$ is nonincreasing in x: by Lemma 4.7.6 of [4], $Tv(x,q)$ will be submodular. We have the decomposition:

$$\Delta_q Tv(x)=c(x,1)-c(x,0)+\theta\Delta_q\hat{T}v(x)\ , \tag{4}$$

$$\text{where: } \Delta_q\hat{T}v(x)=\mathbb{E}v(S((x-1)^+)+A)\ -\ \mathbb{E}v(S(x)+A)\ . \tag{5}$$

Using (2), the difference $c(x,1)-c(x,0)$ is seen to be nonincreasing in $x \geq 0$. We then prove the nonincreasingness of $x \mapsto \Delta_q \hat{T} v(x)$ for any $x > 0$. In that case, we use the stochastic decomposition $S(x) = S(1) + S(x-1)$ (where the random variables in the right-hand side are independent), in (5) to get:

$$\begin{aligned} \Delta_q \hat{T} v(x) &= \mathbb{E} v(S(x-1)+A) \;-\; \mathbb{E} v(S(x-1)+S(1)+A) \\ &= \;-\; \sum_{a,s} \mathbb{P}(A=a, S(1)=s)\; \mathbb{E}\left[u_{a,s}(S(x-1))\right] \;, \end{aligned} \tag{6}$$

where we have defined: $u_{a,s}(y) := v(y+s+a) - v(y+a)$. Since v is increasing and convex, the function $u_{a,s}(y)$ is nonnegative and increasing for all nonnegative values of a and s. The stochastic increasingness of the $S(x)$ (Proposition 1), implies that $\mathbb{E} u_{a,s}(S(x)) \geq \mathbb{E} u_{a,s}(S(x-1))$, for all $x \geq 1$ and all s, $a \geq 0$. This last inequality is conserved by convex combinations. As a result, the expression (6) is a nonincreasing function of $x > 0$. It is also negative, so that when $x = 1$:

$$\Delta_q T v(1) \;=\; c_B - c_C + \theta \Delta_q \hat{T} v(1) \;\leq\; c_B \;=\; \Delta_q T v(0) \;.$$

The function is therefore nonincreasing at $x = 0$ as well.

Lemma 3. *Let $\tilde{v}$ be the function defined by $\tilde{v}(x) = \min_q Tv(x,q)$ for any $x \in \mathbb{N}$. Then $\tilde{v}$ is nondecreasing convex in x if v is nondecreasing convex in x.*

The proof involves a case-by-case analysis, based on the fact that the function $q_y^* := \arg\min_q Tv(y,q)$ is decreasing, a consequence of Lemma 2 and 4.7.1 of [4].

3.2 Structural Properties of the Optimal Policy

One calls a threshold policy (sometimes, "control limit policy") a policy such that $q(x) = q_1$ if $x < \nu$ and $q(x) = q_2$ if $x \geq \nu$, where q_1 and q_2 are in $\mathcal{Q}$ and ν is called the threshold. For our problem, $q_1 = 0$, $q_2 = 1$ and an infinite threshold means that it is never optimal to accept customers.

Theorem 1. *The optimal policy is increasing in x (it is a monotone control) and is a threshold policy.*

The proof is based on Theorem 6.11.3 of [4]. First of all, technical issues related to the unboundedness of the cost function have to be checked. Next, the theorem is applied with V^σ the set of nondecreasing convex functions, and $\mathcal{D}^\sigma$ the set of monotone controls. Lemmas 1– 3 combined with Lemma 4.7.1 of [4] prove that the class of functions V^σ is preserved by the stochastic programming operator. Therefore, there exists an optimal policy which is a monotone control. Given that the action space has two elements, this is actually a threshold policy.

3.3 The Optimal Threshold

The optimal threshold can actually be computed explicitly:

Theorem 2. *Let $\psi = c_B - c_C/(1-\overline{\alpha}\theta)$. Then: a) if $\psi > 0$, the optimal threshold is $\nu = +\infty$; b) if $\psi < 0$, the optimal threshold is $\nu = 1$; c) $\psi = 0$, any threshold policy $\nu \geq 1$ gives the same value.*

As a first step in the proof, a direct computation provides the following expression for the value of the threshold policy with parameter ν:

$$V_\nu(x) = \frac{c_C}{1-\theta\overline{\alpha}} \left(x \ + \ \frac{\theta \mathbb{E}(A)}{1-\theta} \right) \ + \ \psi \left(\sum_{n=0}^{\infty} \theta^n \mathbb{P}(R_\nu^{(n)}(x) \geq \nu) \right) .$$

Then the function $\Phi_\nu(x,\theta)$, defined as the series in the above equation, is shown to be positive, increasing with respect to x for every fixed ν and decreasing with respect to ν for fixed x. The proof of this relies on a sample path comparison argument. The dependence on ν being concentrated in the function Φ_ν, the minimum is either at $\nu = 1$ or $\nu = +\infty$, depending on the sign of ψ.

4 Conclusions

In this paper we show that the optimal control of service in a single-server queue with impatience is a threshold policy and we give the value of this threshold. If the framework used is not original, its application here requires some additional concepts which do not appear in previous works. For example, proving here the monotonicity of the control requires a convex value function contrarily to the usual cases where only monotonicity of the value function is required (see [4]). On the other hand, the simplicity of the result raises the idea that a proof not using the "structural" framework should exist. We discuss this issue in more detail in [2]. In particular, we explain why the exchange arguments usually invoked to compare policies, do not apply to our case.

The extension of the problem to the case where the server may serve more than one customer at a time, does not work in a straightforward manner. Actually, starting with $B = 2$, the value function of the problem ceases, in general, to have the submodularity property required in Lemma 2. On the other hand, no experimental evidence has contradicted, so far, the possibility that the optimal control still be of threshold type. The challenge of further research on the topic will therefore be to find the appropriate properties that can be propagated by the dynamic programming operator in this case.

References

1. Deb, R.K., Serfozo, R.F.: Optimal control of batch service queues. Adv. App. Prob. 5(2), 340–361 (1973)
2. Hyon, E., Jean-Marie, A.: Scheduling in a queuing system with impatience and setup costs. Tech. Rep. RR-6881, version 2, INRIA (Feb 2010)
3. Papadaki, K.P., Powell, W.B.: Exploiting structure in adaptative dynamic programming algorithms for a stochastic batch service problem. EJOR 142, 108–127 (2002)
4. Puterman, M.: Markov Decision Processes Discrete Stochastic Dynamic Programming. Wiley (2005)