A Nearly-Optimal Index Rule for Scheduling of Users with Abandonment

Urtzi Ayesta*†, Peter Jacko* and Vladimir Novak*‡

*BCAM — Basque Center for Applied Mathematics, 48170 Bilbao, Spain
Email: {ayesta, jacko} @ bcamath.org
†IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain
‡Comenius University, 84248 Bratislava, Slovakia
Email: novakvlado @ gmail.com

***Abstract*—We analyze a comprehensive model for multi-class job scheduling accounting for user abandonment, with the objective of minimizing the total discounted or time-average sum of linear holding costs and abandonment penalties. We assume geometric service times and Bernoulli abandonment probabilities. We solve analytically the case in which there are 1 or 2 users in the system to obtain an optimal index rule. For the case with more users we use recent advances from the restless bandits literature to obtain a new simple index rule, denoted by AJN, which we propose to use also in the system with arrivals. In the problem without abandonment, the proposed rule recovers the $c\mu$-rule which is well-known to be optimal both without and with arrivals. Under certain conditions, our rule is equivalent to the $c\mu/\theta$-rule, which was recently proposed and shown to be asymptotically optimal in a multi-server system with overload conditions. We present results of an extensive computational study that suggest that our rule is almost always superior or equivalent to other rules proposed in the literature, and is often optimal.**

I. Introduction

Abandonment (aka reneging) is an ubiquitous phenomena that happens in a multitude of systems, for instance, customers can abandon after being waiting for too long in a queue, or users in the Internet may give up a transfer if the connection is slow. User abandonment has a very negative impact from the performance point of view both from the user's and the system's perspective. A user who abandons considers that the system is poorly managed or not well dimensioned, and will therefore get a bad impression. From the system's point of view the abandonment might imply that resources have been wasted by allocating resources to a user that decided to abandon anyway. For this reason it is important to design efficient scheduling policies in the presence of abandonment. Mathematically, abandonments can be studied by queueing models, but as a consequence of the complexity, the problem of how to schedule impatient users is not completely understood. This is the main focus of the present paper where based on recent developments on the theory of multi-armed bandits we derive a simple implementable scheduling rule for multi-class systems that shows a nearly-optimal performance.

Research partially supported by grant MTM2010-17405 of the MICINN (Spain) and grant PI2010-2 of the Department of Education and Research (Basque Government). The work of V. Novak was carried out thanks to the BCAM internship program and to the "ZA EFM" Alumni association (Slovakia).

The literature on models that incorporate user abandonments is rich, and there has been a surge in recent years motivated by their application on health-care, call-centers and the Internet. An important stream of works investigate the performance of systems in the presence of abandonments (see for example [1], [2], [3], [4], [5]). Abandonment systems have also been looked at from a game-theoretic perspective (see [6, Chapter 5] and references therein). More relevant to our work are the papers which deal with the control of systems in the presence of abandonments (see, for example, [7], [8], [9], [10], [11].) In [7] the authors consider three different models, and derive heuristic rules for each of them. [8] studies how to schedule optimally users in a patient triage problem. [9] deals with a two-class system and derives sufficient conditions in order priority rules to be optimal. In [10] the authors derive scheduling policies for call centers with two classes of customers. The most relevant paper for our work is [11], where the authors investigate a multi-server model with multiple classes and introduce the $c\mu/\theta$-rule. The $c\mu/\theta$-rule gives service to the class k with highest $c_k\mu_k/\theta_k$, where c_k is the holding cost rate, μ_k is the service rate and θ_k is the abandonment rate. It is shown in [11] that the $c\mu/\theta$ rule minimizes asymptotically (in the fluid limit sense) the time-average holding cost in the overload case.

In this paper we aim at solving the problem of scheduling users in a system with abandonments where the objective is to minimize the total discounted or time-average cost (holding costs and abandonment penalties). In disparity with [11], the user in service also adds to the cost function, which as we will see, has a significant consequence. We formulate the problem as a discrete time Markov Decision Process (MDP). Service times are geometrically distributed, and we assume that in every time step, every user (except for the one in service, if any) may abandon the system independently of everything else. Every user in the system incurs a holding cost, and an abandonment cost in case of abandonment. Thus, user k's behavior is completely characterized by the mean service time $1/\mu_k$, the abandonment probability θ_k, the holding cost c_k and the abandonment cost d_k. This formulation

can be seen as a generalization of the job sequencing problem with geometrically distributed service times formulated in [12]. This modeling framework is known as the multi-armed restless bandit problem, an optimization problem extremely difficult and proven PSPACE-hard [13], that is, its complexity grows exponentially in time and in memory requirements. The term restless refers to the fact that, as a consequence of abandonment, the state of all users in the system varies in time regardless on whether they are served or not. Restless problems can be solved analytically only in a few cases (typically with largely restricted dynamics), and this explains to some extent why optimality results on scheduling in systems with abandonment are so scarce in queueing theory.

We solve analytically the case in which there are one or two users in the system and characterize in closed form the switching curves which give rise to an index policy. To solve the case with more than two users, we follow the approach of Whittle [14] that allows us to derive a nearly optimal scheduling rule by calculating the indices as described in [15]. The main idea is to relax the sample path constraint (that imposes that only one user is served at a time) by letting the average/discounted number of users served in a slot to be one. This relaxation simplifies significantly the problem. The optimal policy of the relaxed formulation becomes now of index type (as in the classical multi-armed bandit problem), that is, we can calculate for each user certain index called *price* (that depends only on the user's parameters and on whether she is still waiting or not), and the optimal scheduler serves in every slot the users with actual index higher than a threshold (the value of the threshold ensures that the average/discounted number of users served in a slot is precisely one).

We prove in this paper that the value of the price is

$$\frac{c_k(\mu_k-\theta_k)+d_k\theta_k(1-\beta+\beta\mu_k)}{1-\beta+\beta\theta_k}, \tag{1}$$

where $0 \leq \beta \leq 1$ is the discount factor. The optimal policy for the relaxed problem need not be feasible for the original problem, but allows to construct a heuristic index policy for the original problem by serving the user with currently highest price, we call this rule the *AJN-rule*. This heuristic rule is feasible (only one user is served at a time) and is typically reported to have an extremely good performance [16]. In addition, it was shown in [17] that index policies approach optimality as the number of users grows to infinite (if certain additional assumptions hold).

Under the time-average criterion (i.e., when $\beta = 1$), the AJN-rule serves the class with highest value

$$\frac{c_k(\mu_k-\theta_k)+d_k\theta_k\mu_k}{\theta_k} \tag{2}$$

We observe that there is a fundamental difference between the $c\mu/\theta$-rule and the AJN-rule in this case. With $c\mu/\theta$, the server always chooses serving rather than idling, whereas under AJN, the server might prefer to idle if the value of θ is sufficiently large (there is no benefit in serving a user with a very large abandonment probability). The difference is explained by the different objective criteria adopted in the two models. In [11] the objective was the number of jobs in the queue. In that case it is always profitable to serve a user (since the user will immediately stop contributing to the cost), that is, there is no trade-off between serving and idling. In the case without abandonment, the $c\mu$-rule is optimal for both criteria, which is explained by the fact that the system is work conserving (the probability that the server is busy is the same for all policies). However, in the presence of user abandonments, the system is no longer work-conserving, and both criteria give very different solutions. Thus, we believe that in the case of not work-conserving systems, much care is needed in defining the objective criteria.

We perform computational experiments for a wide variety of scenarios and we compare the performance of various scheduling policies. In the experiments section we consider a more realistic scenario where users arrive randomly according to a Poisson process. Even though AJN is obtained by solving a model without arrivals, existing literature gives strong evidence to support the claim that AJN may perform very well also in the presence of new arrivals, particularly if the arrival process is Bernoulli or Poisson. In fact it has been shown in a wide variety of models that the optimal scheduling policy with a fixed number of users is also optimal in the case of arrivals (see for example [18] and [19, Theorem 3.28] for the M/G/1 queue, [20] and [21], [22] for the $c\mu$-rule, [23] for a single server queue with feedback and [24], [19] and [25] for the multi-armed bandit problem.) In fact, AJN and $c\mu/\theta$ may be equivalent if the holding costs are the same for all the classes. However, when there is class of users with θ larger than μ, or when c's differ across classes, then our numerical results illustrate that in the overload case the performance of the AJN can significantly outperform $c\mu/\theta$.

The rest of the paper is organized as follows. In Section II we present the problem description. Section III contains the formulation of the problem as a Markov Decision Process. In Section IV we solve analytically the problem when there are one or two users. In Section V we introduce the relaxed formulation of the original problem with any number of users, since that is intractable for finding an optimal solution. Section VI contains the main contribution of this paper, that is, the analytical resolution of the relaxed problem and the heuristic rule for the original stochastic optimization formulation. Section VII presents the numerical experiments. Proofs are not presented in whole detail due to space constraints.

II. Problem Description

In this paper we analyze the multi-class job scheduling problem, in which we allow for abandonment due to users' mobility or impatience. Consider $K-1$ jobs waiting for service of a server that can serve one job at a time. The service of job k is completed (if being served) with probability $\mu_k > 0$ and the probability of her abandonment (if not being served) is $\theta_k \geq 0$. We assume that the user in service cannot abandon.

Thus, the jobs (i.e., users) are assumed independent of each other.

Let $c_k > 0$ be the holding cost incurred for user k waiting in the queue. Further, let $d_k > 0$ be the abandonment penalty incurred for user k if she abandons the system without having her job completed. If the server is allocated to a user whose job has already been completed, then no service occurs.

We incorporate the following parameter that makes the problem extensively flexible to incorporate additional conditions or options, and turns out to be crucial for creating a not work-conserving system. It is allowed to allocate the server to an alternative task (such as idling, battery recharging or service maintenance), for which we obtain an *alternative-task reward* κ. For instance, the role of this alternative task with a positive κ could be to turn off the server allocation when all the users have too high abandonment rates. On the other hand, by setting this parameter to a negative value we may force the server to be non-idling (whenever there are waiting jobs), or it can be simply set to zero narrowing the focus to the classic problem.

The joint goal is to minimize the expected aggregate holding and abandonment costs minus the alternative-task reward, over an infinite horizon. The server is assumed to be preemptive (i.e., the service of a job or the alternative task can be interrupted at any moment even if not completed). Thus, the server continuously decides to which user (if any) it should be allocated.

A. Variant without Abandonment

If $\theta_k = 0$ for all k (i.e., there is no abandonment) and $\kappa = 0$, then this problem recovers the classic job scheduling problem considered in [12], [20], [21], [22], for which the following greedy rule attains such a goal:

Rule 1 ($c\mu$-rule): Allocate the server to any waiting job of the non-empty class with the highest value $c_k\mu_k$.

For a given class k, $c_k\mu_k$ measures the expected savings in holding costs, or the efficiency of attaining the goal, if user of class k is served. Thus, the $c\mu$-rule allocates the server to the user who contributes most efficiently to minimization of the expected aggregate holding cost.

III. MDP Formulation

Since the $c\mu$-rule is optimal both under general arrival distribution and under no arrivals, and both in continuous-time and discrete-time model, we set out to analyze the discrete-time model without arrivals, in order to obtain a rule accounting for abandonment whose performance in the continuous-time model with arrivals we later evaluate by means of numerical experiments. We set the model in the framework of the dynamic and stochastic resource allocation problem and follow the approach to design prices as described in [26].

Consider the time slotted into epochs $t \in \mathcal{T} := \{0, 1, 2, \dots\}$ at which decisions can be made. The time epoch t corresponds to the beginning of the time period t. Suppose that at $t = 0$ there are $K-1 \geq 1$ users awaiting service from the server that at each epoch chooses (at most) one of the users to serve. If no user is chosen, then the server is allocated to the alternative task, i.e., there are K competing options, labeled by $k \in \mathcal{K}$. Thus, the server is allocated to exactly one option at a time.

A. Jobs and Users

Every user $k = 1, 2, \dots, K-1$ can be allocated either zero or full capacity of the server. We denote by $\mathcal{A} := \{0, 1\}$ the *action space*, i.e., the set of allowable levels of capacity allocation. Here, action 0 means allocating zero capacity (i.e., "not serving"), and action 1 means allocating full capacity (i.e., "serving"). This action space is the same for every user k.

Each job/user k is defined independently of other jobs/users as the tuple

$$\left(\mathcal{N}_k, (\boldsymbol{W}_k^a)_{a\in\mathcal{A}}, (\boldsymbol{R}_k^a)_{a\in\mathcal{A}}, (\boldsymbol{P}_k^a)_{a\in\mathcal{A}}\right),$$

where

- $\mathcal{N}_k := \{0, 1\}$ is the *state space*, where state 0 represents a job already completed or abandoned, and state 1 means that the job is uncompleted and not abandoned;
- $\boldsymbol{W}_k^a := \left(W_{k,n}^a\right)_{n\in\mathcal{N}_k}$, where $W_{k,n}^a$ is the (expected) one-period capacity consumption, or *work* required by user k at state n if action a is decided at the beginning of a period; in particular, for any $n \in \mathcal{N}_k$,
$$W_{k,n}^1 := 1, \qquad W_{k,n}^0 := 0;$$
- $\boldsymbol{R}_k^a := \left(R_{k,n}^a\right)_{n\in\mathcal{N}_k}$, where $R_{k,n}^a$ is the expected one-period *reward* earned by user k at state n if action a is decided at the beginning of a period; in particular,
$$R_{k,0}^1 := 0, \quad R_{k,1}^1 := -c_k \cdot (1-\mu_k) + 0 \cdot \mu_k,$$
$$R_{k,0}^0 := 0, \quad R_{k,1}^0 := -c_k \cdot (1-\theta_k) - d_k \cdot \theta_k;$$
- $\boldsymbol{P}_k^a := \left(p_{k,n,m}^a\right)_{n,m\in\mathcal{N}_k}$ is the user-k stationary one-period *state-transition probability matrix* if action a is decided at the beginning of a period, i.e., $p_{k,n,m}^a$ is the probability of moving to state m from state n under action a; in particular, we have

$$\boldsymbol{P}_k^1 := \begin{array}{c} \\ 0 \\ 1 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 1 & 0 \\ \mu_k & 1-\mu_k \end{pmatrix} \end{array}, \quad \boldsymbol{P}_k^0 := \begin{array}{c} \\ 0 \\ 1 \end{array}\!\!\begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 1 & 0 \\ \theta_k & 1-\theta_k \end{pmatrix} \end{array}.$$

The dynamics of user k is thus captured by the *state process* $X_k(\cdot)$ and the *action process* $a_k(\cdot)$, which correspond to state $X_k(t) \in \mathcal{N}_k$ and action $a_k(t) \in \mathcal{A}$ at all time epochs $t \in \mathcal{T}$. As a result of deciding action $a_k(t)$ in state $X_k(t)$ at time epoch t, the user k consumes the allocated capacity, earns the reward, and evolves its state for the time epoch $t+1$.

Note that we have the same action space $\mathcal{A}$ available at every state, which assures a technically useful property that

$\boldsymbol{W}_k^a, \boldsymbol{R}_k^a, \boldsymbol{P}_k^a$ are defined in the same dimensions under any $a \in \mathcal{A}$. Note also that state 0 is absorbing.

B. Alternative Task

We model the alternative task as a *static κ-user* with a single state 0 and with reward κ if served, i.e., such a user $k = K$ is defined by $\mathcal{N}_K := \{0\}, W_{K,0}^a := a, R_{K,0}^a := \kappa a, p_{K,0,0}^a := 1$ for all $a \in \mathcal{A}$.

C. A Unified Optimization Criterion

Before describing the problem we first define an averaging operator that will allow us to discuss the infinite-horizon problem under the traditional β-discounted criterion and the time-average criterion in parallel. Let $\Pi_{X,a}$ be the set of all the policies that for each time epoch t decide (possibly *randomized*) action $a(t)$ based only on the state-process history $X(0), X(1), \ldots, X(t)$ and on the action-process history $a(0), a(1), \ldots, a(t-1)$ (i.e., *non-anticipative*). Let $\mathbb{E}_\tau^\pi$ denote the expectation over the state process $X(\cdot)$ and over the action process $a(\cdot)$, conditioned on the state-process history $X(0), X(1), \ldots, X(\tau)$ and on policy π.

Consider any expected one-period quantity $Q_{X(t)}^{a(t)}$ that depends on state $X(t)$ and on action $a(t)$ at any time epoch t. For any policy $\pi \in \Pi_{X,a}$, any initial time epoch $\tau \in \mathcal{T}$, and any *discount factor* $0 \leq \beta \leq 1$ we define the infinite-horizon *β-average quantity* as[1]

$$\mathbb{B}_\tau^\pi \left[Q_{X(\cdot)}^{a(\cdot)}, \beta, \infty \right] := \lim_{T \to \infty} \frac{\sum_{t=\tau}^{T-1} \beta^{t-\tau} \, \mathbb{E}_\tau^\pi \left[Q_{X(t)}^{a(t)} \right]}{\sum_{t=\tau}^{T-1} \beta^{t-\tau}}. \quad (3)$$

The β-average quantity recovers the traditionally considered quantities in the following three cases:

- *expected time-average quantity* when $\beta = 1$.
- *expected total β-discounted quantity*, scaled by constant $1 - \beta$, when $0 < \beta < 1$;
- *myopic quantity* when $\beta = 0$.

Thus, when $\beta = 1$, the problem is formulated under the *time-average criterion*, whereas when $0 < \beta < 1$ the problem is considered under the *β-discounted criterion*. The remaining case when $\beta = 0$ reduces to a static problem and hence is considered in order to define a *myopic policy*. In the following we consider the discount factor β to be fixed and the horizon to be infinite, therefore we omit them in the notation and write briefly $\mathbb{B}_\tau^\pi \left[Q_{X(\cdot)}^{a(\cdot)} \right]$.

D. Optimization Problem

We now describe in more detail the problem we consider. Let $\Pi_{\boldsymbol{X},\boldsymbol{a}}$ be the space of randomized and non-anticipative policies depending on the joint state-process $\boldsymbol{X}(\cdot) := (X_k(\cdot))_{k \in \mathcal{K}}$ and deciding the joint action-process $\boldsymbol{a}(\cdot) := (a_k(\cdot))_{k \in \mathcal{K}}$, i.e., $\Pi_{\boldsymbol{X},\boldsymbol{a}}$ is the *joint policy space*.

[1]For definiteness, we consider $\beta^0 = 1$ for $\beta = 0$.

For any discount factor β, the problem is to find a joint policy $\boldsymbol{\pi}$ maximizing the *objective* given by the β-average aggregate reward starting from the initial time epoch 0 subject to the family of *sample path* allocation constraints, i.e.,

$$\max_{\boldsymbol{\pi} \in \Pi_{\boldsymbol{X},\boldsymbol{a}}} \mathbb{B}_0^{\boldsymbol{\pi}} \left[\sum_{k \in \mathcal{K}} R_{k,X_k(\cdot)}^{a_k(\cdot)} \right] \quad \text{(P)}$$

$$\text{subject to } \mathbb{E}_t^{\boldsymbol{\pi}} \left[\sum_{k \in \mathcal{K}} a_k(t) \right] = 1, \text{ for all } t \in \mathcal{T}$$

Note that the constraint could equivalently be expressed in words as that for all $t \in \mathcal{T}$: $\sum_{k \in \mathcal{K}} a_k(t) = 1$ under policy π and for any possible joint state-process history $\boldsymbol{X}(0), \boldsymbol{X}(1), \ldots, \boldsymbol{X}(t)$.

IV. Special Cases

Problem (P) is hard to solve in the whole generality, but we have identified special cases that admit an analytical solution of the Bellman equation, summarized in this section. Surprisingly, to the best of our knowledge no one has optimally solved the cases of one or two users before.

In the case of a single user competing with the alternative task, we introduce the following index (1U):

$$\nu_k^{1\mathrm{U}} := c_k(\mu_k - \theta_k) + d_k \theta_k (1 - \beta + \beta \mu_k). \quad (4)$$

Proposition 1: The following holds for problem (P) with $K = 2$ and the alternative task reward $\kappa = 0$:

(i) If $\nu_1^{1\mathrm{U}} \geq 0$, then it is optimal to serve user 1;
(ii) If $\nu_1^{1\mathrm{U}} \leq 0$, then it is optimal to allocate the server to the alternative task ($k = 2$).

Proof: There are only two possible states of the system: (0) meaning that the system is empty, and (1), meaning that there is a single user of class 1. The Bellman equation gives the optimal expected total β-discounted values $V_{(0)}^* = 0$, and

$$V_{(1)}^* = \max\{R_{1,1}^1 + \beta p_{1,1,1}^1 V_{(1)}^*; R_{1,1}^0 + \beta p_{1,1,1}^0 V_{(1)}^*\},$$

where the first term refers to serving the user and the second term to idling. After plugging the definitions into these two terms, we solve the Bellman equation and evaluate the value function assuming that serving is optimal:

$$V_{(1)}^* = \frac{-c_1(1 - \mu_1)}{1 - \beta + \beta \mu_1}.$$

Further, we obtain that serving the user is better than or equivalent to idling if

$$\left(-\beta V_{(1)}^* + c_1 \right) (\mu_1 - \theta_1) + d_1 \theta_1 \geq 0.$$

Putting the last equality together with this inequality then yields (i). Claim (ii) is obtained analogously. ∎

In the case of two users competing among themselves (2U), due to the technical complexity of the problem we concentrate only on the undiscounted case ($\beta = 1$). We introduce the following index for users $k = 1, 2$ with respect to the other

user:

$$\nu_k^{2\mathrm{U}} := \frac{c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k}{\mu_k[1-(1-\mu_{3-k})(1-\theta_k)]}. \tag{5}$$

Notice that it depends on the other user's parameters, but only through the service rate μ_{3-k}.

Proposition 2: Suppose that $\nu_k^{1\mathrm{U}} \geq 0$ for $k = 1, 2$. The following holds for problem (P) with $K = 3$, $\beta = 1$, and the alternative task reward $\kappa = 0$:

(i) If $\nu_1^{2\mathrm{U}} \geq \nu_2^{2\mathrm{U}}$, then it is optimal to serve user 1;
(ii) If $\nu_1^{2\mathrm{U}} \leq \nu_2^{2\mathrm{U}}$, then it is optimal to serve user 2.
(iii) It is optimal to allocate the server to the alternative task if and only if $\nu_1^{1\mathrm{U}} = \nu_2^{1\mathrm{U}} = 0$.

Proof: The proof goes along the same lines as the one above, but is significantly longer and is omitted due to space limitations. ∎

V. Relaxations and Decomposition

For larger values of K the problem is most likely analytically intractable, and therefore we approach it in an alternative way.

A. Relaxations

For notational reasons we will use the fact that $W_{k,X_k(t)}^{a_k(t)} = a_k(t)$ (cf. definitions in Section III) and instead of the constraints in (P) we will consider the sample path *consumption* constraints $\mathbb{E}_t^{\boldsymbol{\pi}}\left[\sum_{k\in\mathcal{K}} W_{k,X_k(t)}^{a_k(t)}\right] = 1$, for all $t \in \mathcal{T}$. These constraints imply the *epoch-t expected* consumption constraints,

$$\mathbb{E}_0^{\boldsymbol{\pi}}\left[\sum_{k\in\mathcal{K}} W_{k,X_k(t)}^{a_k(t)}\right] = 1, \text{ for all } t \in \mathcal{T} \tag{6}$$

requiring that the capacity be fully allocated at every time epoch if conditioned on $\boldsymbol{X}(0)$ only. Finally, we may require this constraint to hold only on β-average, as the *β-average capacity consumption constraint*

$$\mathbb{B}_0^{\boldsymbol{\pi}}\left[\sum_{k\in\mathcal{K}} W_{k,X_k(\cdot)}^{a_k(\cdot)}\right] = \mathbb{B}_0^{\boldsymbol{\pi}}[1]. \tag{7}$$

Using $\mathbb{B}_0^{\boldsymbol{\pi}}[1] = 1$, we obtain the following *relaxation* of problem (P),

$$\max_{\boldsymbol{\pi}\in\Pi_{\boldsymbol{X},\boldsymbol{a}}} \mathbb{B}_0^{\boldsymbol{\pi}}\left[\sum_{k\in\mathcal{K}} R_{k,X_k(\cdot)}^{a_k(\cdot)}\right] \tag{P$^{\mathrm{W}}$}$$
$$\text{subject to } \mathbb{B}_0^{\boldsymbol{\pi}}\left[\sum_{k\in\mathcal{K}} W_{k,X_k(\cdot)}^{a_k(\cdot)}\right] = 1.$$

This relaxation was introduced in [14]. The above arguments thus provide a proof of the following result.

Proposition 3: Problem (P$^{\mathrm{W}}$) is a relaxation of problem (P).

The *Whittle relaxation* (P$^{\mathrm{W}}$) can be approached by traditional Lagrangian methods, introducing a Lagrangian parameter, say ν, to dualize the constraint, obtaining thus the following Lagrangian relaxation,

$$\max_{\boldsymbol{\pi}\in\Pi_{\boldsymbol{X},\boldsymbol{a}}} \mathbb{B}_0^{\boldsymbol{\pi}}\left[\sum_{k\in\mathcal{K}} R_{k,X_k(\cdot)}^{a_k(\cdot)} - \nu\sum_{k\in\mathcal{K}} W_{k,X_k(\cdot)}^{a_k(\cdot)}\right] + \nu. \tag{P$_\nu^{\mathrm{L}}$}$$

The classic Lagrangian result says the following:

Proposition 4: For any ν, problem (P$_\nu^{\mathrm{L}}$) is a relaxation of problem (P$^{\mathrm{W}}$), and further a relaxation of problem (P).

Note finally that by the definition of relaxation, (P$_\nu^{\mathrm{L}}$) for every ν provides an upper bound for the optimal value of both problem (P$^{\mathrm{W}}$) and problem (P).

B. Decomposition into Single-User Subproblems

We now set out to decompose the optimization problem (P$_\nu^{\mathrm{L}}$) as it is standard for Lagrangian relaxations, considering ν as a parameter. Notice that any joint policy $\boldsymbol{\pi} \in \Pi_{\boldsymbol{X},\boldsymbol{a}}$ defines a set of single-user policies $\widetilde{\pi}_k$ for all $k \in \mathcal{K}$, where $\widetilde{\pi}_k$ is a randomized and non-anticipative policy depending on the *joint* state-process $\boldsymbol{X}(\cdot)$ and deciding the *user-k* action-process $a_k(\cdot)$. We will write $\widetilde{\pi}_k \in \Pi_{\boldsymbol{X},a_k}$. We will therefore study the user-k subproblem

$$\max_{\widetilde{\pi}_k\in\Pi_{\boldsymbol{X},a_k}} \mathbb{B}_0^{\widetilde{\pi}_k}\left[R_{k,X_k(\cdot)}^{a_k(\cdot)} - \nu W_{k,X_k(\cdot)}^{a_k(\cdot)}\right]. \tag{8}$$

VI. Solution

In this section we will identify a set of optimal policies $\widetilde{\pi}_k^*$ to (8) for all users k, and using them we will construct a joint policy $\boldsymbol{\pi}$ that is feasible but not necessarily optimal for problem (P).

A. Optimal Solution to Single-User Subproblem via Prices

Problem (8) falls into the framework of *restless bandits* and can be optimally solved by assigning a set of prices $\nu_{k,n}$ to each state $n \in \mathcal{N}_k$ under certain conditions [16].

Let us denote for user $k \leq K-1$, $\nu_{k,0}^{\mathrm{AJN}} := 0$, and

$$\nu_{k,1}^{\mathrm{AJN}} := \frac{c_k(\mu_k - \theta_k) + d_k\theta_k(1-\beta+\beta\mu_k)}{1-\beta+\beta\theta_k}, \tag{9}$$

and for the alternative task $k = K$, $\nu_{K,0}^{\mathrm{AJN}} := \kappa$. Then we can prove the following result.

Proposition 5: For problem (8) and all $k \leq K-1$, supposing that $\nu_k^{1\mathrm{U}} \geq 0$, the following holds:

(i) if $\nu \leq \nu_{k,1}^{\mathrm{AJN}}$, then it is optimal to serve waiting user k;
(ii) if $\nu \geq \nu_{k,1}^{\mathrm{AJN}}$, then it is optimal not to serve waiting user k;
(iii) if $\nu \leq \nu_{k,0}^{\mathrm{AJN}}$, then it is optimal to serve job k when it is already completed or abandoned;
(iv) if $\nu \geq \nu_{k,0}^{\mathrm{AJN}}$, then it is optimal not to serve job k when it is already completed or abandoned;
(v) if $\nu \leq \nu_{K,0}^{\mathrm{AJN}}$, then it is optimal to allocate the server to the alternative task K;
(vi) if $\nu \geq \nu_{K,0}^{\mathrm{AJN}}$, then it is optimal not to allocate the server to the alternative task K.

Proof: The proof of this proposition is based on establishing indexability of the problem and computing the

TABLE I
PARAMETERS FOR ALL THE SCENARIOS IN COMPUTATIONAL EXPERIMENTS.

	μ_1	μ_2	θ_1	θ_2	c_1	c_2	d_1	d_2	λ_1	λ_2
Scenario 1	0.7	0.3	$[0,2]$	0.2	1	1	1	1	1	1
Scenario 2	0.4	0.59	$[2,3]$	4	1	1	1	1	1	1
Scenario 3	0.8	0.7	1.2	2.7	1	1	$[0,50]$	1	1	1
Scenario 4	0.8	0.7	1.2	2.7	1	1	$[0,10]$	5	1	1
Scenario 5	0.4	0.1	0.5	0.8	1	$[0.01,20]$	1	1	1	1
Scenario 6	0.4	0.22	0.1	0.2	1	$[1,40]$	1	1	1	1
Scenario 7	0.4	0.3	0.001	0.03	1	$[1,60]$	1	1	1	1
Scenario 8	0.1	0.4	0.12	0.1	$[0.01,20]$	1	50	1	1	1

index values following the survey [16]. Indexability is in fact equivalent to existence of the quantities with stated properties, and is valid because any binary-state MDP is indexable. Index computation is more involved and requires additional definitions and notation, therefore is omitted here. ■

B. Optimal Solution to Relaxations

The vector of policies $\boldsymbol{\pi}^* := (\widetilde{\pi}^*_k)_{k\in\mathcal{K}}$ identified in Proposition 5 is formed by mutually independent single-user optimal policies, therefore this vector is an optimal policy to the Lagrangian relaxation ($\mathrm{P}^{\mathrm{L}}_{\nu}$).

Since a finite-state MDP admits an LP formulation using the standard *state-action frequency* variables (as observed in [15]), strong LP duality implies that there exists ν^* (possibly depending on the joint initial state) such that the Lagrangian relaxation ($\mathrm{P}^{\mathrm{L}}_{\nu^*}$) achieves the same objective value as (P^{W}). Further, if $\nu^* \neq 0$, then LP complementary slackness ensures that the β-average capacity constraint (7) is satisfied by any optimal solution to ($\mathrm{R}^{L}_{\nu^*}$).

C. AJN Rule for Original Problem

Since the original problem requires to allocate the server to exactly one option (one of the users or the alternative task), then at any time epoch t we propose to allocate the server to any option $k^*(t)$ with the highest actual price, i.e.,

$$k^*(t) \in \arg\max_{k\in\mathcal{K}} \nu^{\mathrm{AJN}}_{k,X_k(t)}.$$

Notice that any class without abandonment (i.e., having $\theta_k = 0$) has the index

$$\nu^{\mathrm{AJN}}_{k,1} := \frac{c_k\mu_k}{1-\beta},$$

which is just the $c\mu$ index scaled by a constant. Moreover, under the time-average criterion such a class gets an absolute priority over any class with positive abandonment rate.

Under $\beta = 1$, we obtain the time-average version of the AJN index,

$$\nu^{\mathrm{AJN}}_{k,1} := \frac{c_k(\mu_k-\theta_k) + d_k\theta_k\mu_k}{\theta_k},$$

which we implement in our computational experiments in the next section.

Finally, we just remark that $\beta = 0$ gives rise to the myopic version of the AJN index,

$$\nu^{\mathrm{AJN}}_{k,1} := c_k(\mu_k-\theta_k) + d_k\theta_k.$$

VII. COMPUTATIONAL EXPERIMENTS

In this section we report on an exhaustive study of numerical experiments. We consider a system with two classes of users. Each class is characterized by a set of values for the parameters μ, θ, c and d as before, and the mean rate λ of Poisson arrivals. For higher relevance in applications we consider a continuous-time model (so λ, μ and θ are rates).

We are interested in the time-average performance of the whole system, i.e., including the user in service. Recall that in the case without abandonments, the system is work-conserving which implies that taking or not into account the user in service is equivalent. However, in the case of abandonments these two models are not equivalent, and we believe that the total cost in the system is a more relevant measure. As a consequence, in some cases it may not necessarily be optimal to serve, but rather to idle. This is not captured by the model in [11], where the $c\mu/\theta$-rule was derived, in which only the waiting users were considered. It was shown in [11] that for a non-zero abandonment penalty d, the $c\mu/\theta$-rule is

$$\frac{c_k\mu_k + d_k\theta_k\mu_k}{\theta_k}. \tag{10}$$

We truncate the state space by allowing a maximum number of users in each class, and we then use the uniformization technique in order to obtain a discrete-time representation of the model. Using value iteration [27] we obtain numerically the optimal policy, and we then calculate the relative suboptimality gap produced by the rules AJN, $c\mu/\theta$, $c\mu$ and 2U. We take care that the truncation levels are large enough so that the optimal policy obtained in this way is almost surely the optimal policy in the untruncated problem.

We have investigated a wide range of settings for the parameters in around 200 scenarios, and we report here the results of six representative scenarios in order to provide a global panorama. We further present two additional scenarios with unique and peculiar results. In each of the scenarios, only a single parameter is varied in order to easily depict the

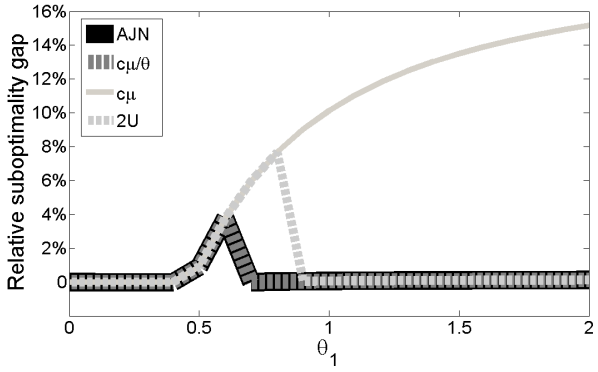

Fig. 1. Relative suboptimality gap in Scenario 1

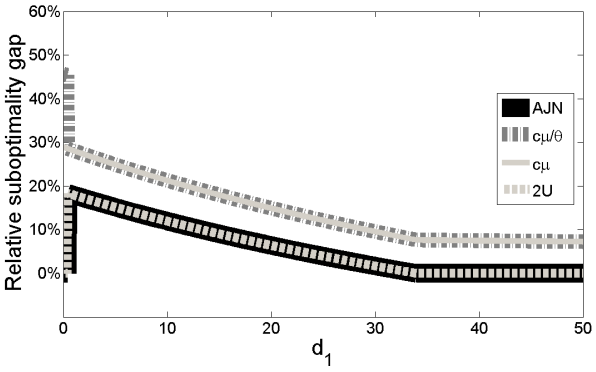

Fig. 3. Relative suboptimality gap in Scenario 3

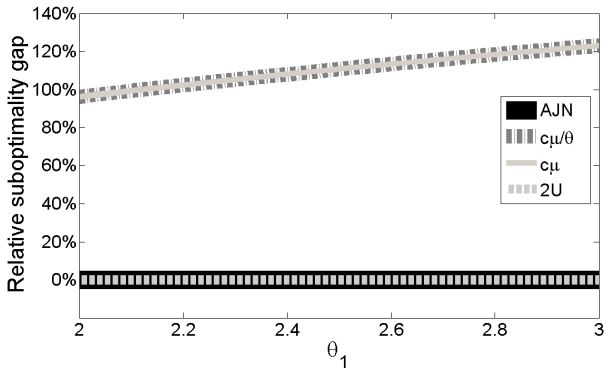

Fig. 2. Relative suboptimality gap in Scenario 2

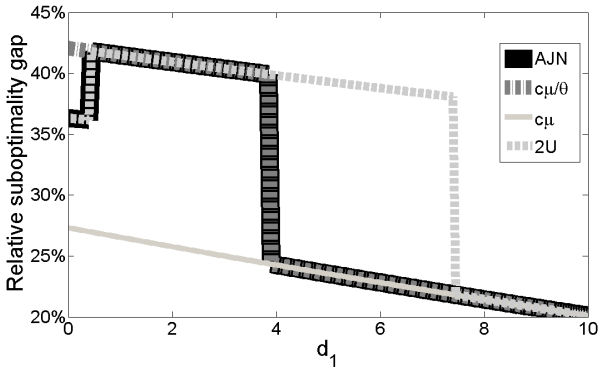

Fig. 4. Relative suboptimality gap in Scenario 4

effect. In Table I we present the parameters considered in each of the scenarios. Note that in all the scenarios the system is in overload (as in [11]), having

$$\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2} > 1,$$

which implies that abandonments are required in order to stabilize the system.

Given the number of parameters that we can choose from, the number of scenarios one can construct is virtually unbounded. Nevertheless, there are some general conclusions that we can draw:

- Almost always the AJN rule is equivalent or outperforms the $c\mu/\theta$ rule;
- In cases in which the optimal policy chooses to idle instead of serving, then AJN is much better than $c\mu/\theta$ or $c\mu$;
- In many scenarios AJN is equal to the optimal policy for almost all values of the varied parameter;
- The switching point of the 2U-rule is often very close to AJN, but usually its suboptimality region is larger.
- If both the 2U and AJN index for class 1 are greater than both for class 2, then it is almost always optimal to serve class 1.

The first six scenarios illustrate these general conclusions.

Scenario 1. (Figure 1) In this scenario the performance of AJN and $c\mu/\theta$ is equivalent, and optimal except for a small interval. We also observe that the $c\mu$ rule performs very poorly. The performance of 2U is quite good, and the only difference with respect to AJN and $c\mu/\theta$ is the switching point where the policy starts serving class-2 users.

Scenario 2. (Figure 2) For this scenario it is optimal not to serve any user, and to let them abandon. The 2U and AJN policies capture this feature (and are optimal), but the $c\mu$ and $c\mu/\theta$ rules do not. As a consequence the performance of the former two is much better than the latter two.

Scenario 3. (Figure 3) For values of d_1 smaller than 35, the optimal policy does not serve any user, and for values larger than 35 it serves class 1 users (or idles if no class 1 user is waiting). All the other policies give priority to class 1, being the only difference that AJN and 2U idle if there is no class 1 user, whereas $c\mu$ and $c\mu/\theta$ serve class 2 in such a case.

Scenario 4. (Figure 4) Compared to the previous scenario, only the value of d_2 is 5 instead of 1, and this produces a significant difference in the results. Even though θ's are still larger than μ's, in this case the performance of 2U and AJN differ during a non-negligible range of values for d_1. Interestingly, the $c\mu$-rule outperforms (or matches) all the other

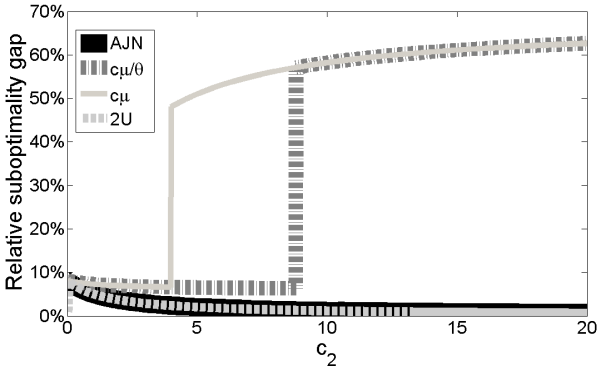

Fig. 5. Relative suboptimality gap in Scenario 5

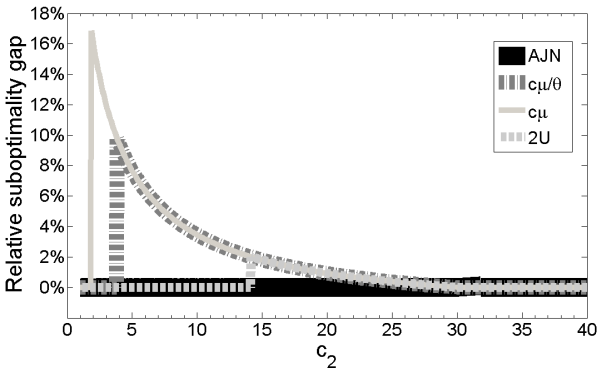

Fig. 6. Relative suboptimality gap in Scenario 6

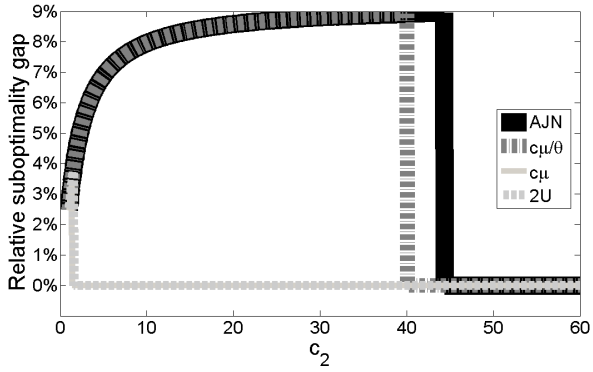

Fig. 7. Relative suboptimality gap in Scenario 7

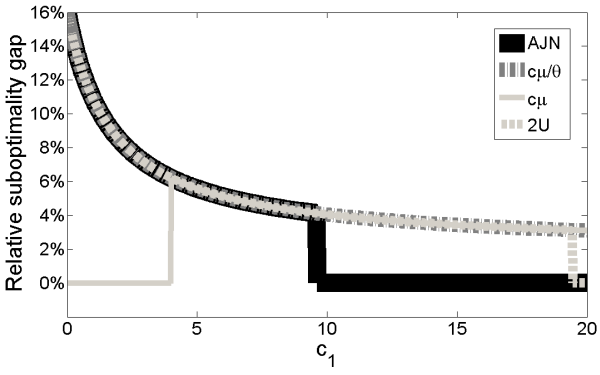

Fig. 8. Relative suboptimality gap in Scenario 8

policies. For this range of values, the optimal policy is not to serve any user. The $c\mu$ rule always serve class 1, and the other policies start serving class 2 users (as a consequence of d_2), and switch to serve class 1 (first AJN together with $c\mu/\theta$, and afterwards 2U) as the value of d_1 becomes larger.

Scenario 5. (Figure 5) The optimal policy is to idle. Policies AJN and 2U give priority to class 1, and to idling if there is no user of class 1 (class 2 is never served). As the value of c_2 increases, the policies $c\mu$ and $c\mu/\theta$ switch to give priority to class 2, what makes a sudden increase in the cost function. The key difference is that AJN depends on the difference $\mu-\theta$, and thanks to this it chooses not to serve class 2 regardless of the value of c_2. AJN's performance is very close to optimal.

Scenario 6. (Figure 6) This is a particularly interesting scenario. Service rates are larger than abandonment rates, but the policy AJN shows a better performance than the other policies. In fact, AJN is optimal for all values of c_2 with the exception of a small range around 32. The optimal policy starts serving class 1 in almost all system states, but as the value of c_2 increases it starts serving class 2 in more joint states. The AJN policy serves class 1 with strict priority for values of c_2 smaller than 32, and class 2 from that moment on. The upward jump for the other indices happens when they start giving priority to class 2 (first $c\mu$ switches, then $c\mu/\theta$ and then 2U).

The following two scenarios illustrate some specific and uncommon phenomena we have found in our experiments.

Scenario 7. (Figure 7) In this scenario the abandonment rate is very small, say negligible. We recall that without abandonments, the $c\mu$-rule is optimal [12], [22]. In the numerical experiments we see that, with a rather surprising exception for $c_2 = 1$, the $c\mu$-rule is indeed optimal in this case and the 2U-rule is equivalent to $c\mu$. Policies AJN and $c\mu/\theta$ start serving class 1, and switch later on to class 2 when the value of c_2 becomes sufficiently large.

We emphasize that this is the only scenario we have found where the decision pattern of 2U and AJN differs completely and also the only one in which $c\mu/\theta$ outperforms AJN (for values of c_2 between 40 and 44).

Scenario 8. (Figure 8) This scenario has a mixed setting since $\mu_1 < \theta_1$, but $\mu_2 > \theta_2$. We vary c_1. The optimal policy always serves class 2 users. For small values of c_1, the $c\mu$-rule serves class 2, but it then switches to class 1, and therefore it is suboptimal for all larger values of c_1. AJN and 2U start serving class 1, and when c_1 becomes sufficiently large they switch to class 2 (since $\mu_1 < \theta_1$).

In this case $c\mu/\theta$ remains suboptimal for all values of c_1.

However, each of the other three policies is optimal on some subrange of the parameter space.

VIII. Conclusion

We have investigated the problem of job scheduling with user abandonments. This is an important problem in several application fields, for which no general solution is known. We have proposed a comprehensive model accounting for both the linear holding costs and abandonment penalties. For the problem with one or two users in the system, we have obtained an optimal solution.

For the more general case with multiple users, we have applied Whittle's relaxation methodology to derive the AJN-rule, a heuristic scheduling rule which has a very simple structure. This rule is under some conditions equivalent to the $c\mu/\theta$-rule that was proven asymptotically optimal in [11]. In many other cases, AJN performs exceptionally well: it is often optimal, and if not, then its suboptimality is small. Numerical results also indicate that in most cases the AJN-rule outperforms the $c\mu/\theta$ and $c\mu$ rules.

The biggest improvement of AJN over $c\mu/\theta$ and $c\mu$ is achieved when it is optimal to idle, that is for instance, when the abandonment probability is large relative to the service-departure probability. In this case the suboptimality of $c\mu/\theta$ can be larger than 100%, while AJN may be optimal at the same time. Another important differences can be observed when the holding costs differ across classes. Interestingly, our scheduling rule recovers known optimal policies in some special cases of the problem, for instance, our rule becomes the $c\mu$-rule if there is no abandonment.

In many cases the performance of 2U is comparable to the performance of AJN. We believe that in a system with more than two classes, our rule should perform increasingly better, as suggested by the asymptotic optimality established for overload conditions. Notice that the 2U-rule cannot be implemented for more than 2 classes, as it depends on the service rate of "the other class".

An important question for future research is to determine under what conditions the AJN-rule is optimal. Based on our numerical experiments, we believe the following hypotheses are true: (1) If AJN index values are sufficiently different across classes, then the AJN-rule is optimal; (2) when the mean number of users in the system is low (perhaps 1 or less), then 2U-rule is optimal; (3) when the mean number of user classes is large, then the AJN-rule is optimal. However, the multi-class problem is much more complex and time- and memory-consuming to simulate (the curse of dimensionality), so these hypotheses could be better verified by analytical results.

References

[1] F. Iravani and B. Balcioğlu, "On priority queues with impatient customers," *Queueing Systems*, vol. 58, pp. 239–260, 2008.

[2] O. Boxma and P. de Waal, "Multiserver queues with impatient customers," in *In Proceedings of ITC-14*, 1994, pp. 743–756.

[3] F. Baccelli, P. Boyer, and G. Hebuterne, "Single-server queues with impatient customers," *Advances in Applied Prabability*, vol. 16, pp. 887–905, 1984.

[4] A. Brandt and M. Brandt, "On the two-class M/M/1 system under preemptive resume and impatience of the prioritized customers," *Queueing Systems*, vol. 47, pp. 147–168, 2004.

[5] P. Brill and M. Posner, "Level crossings in point processes applied to queues: single-server case," *Operations Research*, vol. 25, no. 4, pp. 662–574, 1977.

[6] R. Hassin and M. Haviv, *To Queue or not to Queue: Equilibrium Behavior in Queueing Systems*. Boston etc.: Kluwer Academic Publishers, 2003.

[7] K. Glazebrook, P. Ansell, R. Dunn, and R. Lumley, "On the optimal allocation of service to impatient tasks," *Probability in the Engineering and Informational Sciences*, vol. 41, no. 1, pp. 51–72, 2004.

[8] N. Argon, S. Ziya, and R. Righter, "Scheduling impatient jobs in a clearing system with insights on patient triage in mass-casualty incidents," *Probability in the Engineering and Informational Sciences*, vol. 22, no. 3, pp. 301–332, 2010.

[9] D. Down, G. Koole, and M. Lewis, "Dynamic control of a single server system with abandonments," *Queueing Systems*, vol. 67, no. 1, pp. 63–90, 2011.

[10] O. Jouini, A. Pot, G. Koole, and Y. Dallery, "Online scheduling policies for multiclass call centers with impatient customers," *European Journal of Operational Research*, vol. 207, no. 1, pp. 258–268, 2010.

[11] R. Atar, C. Giat, and N. Shimkin, "The $c\mu/\theta$ rule for many-server queues with abandonment," *Operations Research*, vol. 58, no. 5, pp. 1427-1439, 2010.

[12] D. R. Cox and W. L. Smith, *Queues*. Methuen & Co, 1961.

[13] C. Papadimitriou and J. Tsitsiklis, "The complexity of optimal queueing network," *Mathematics of Operations Research*, vol. 24, no. 2, pp. 293–305, 1999.

[14] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of Applied Probability*, vol. 25, pp. 287–298, 1988.

[15] J. Niño-Mora, "Restless bandits, partial conservation laws and indexability," *Advances in Applied Probability*, vol. 33, no. 1, pp. 76–98, 2001.

[16] ——, "Dynamic priority allocation via restless bandit marginal productivity indices," *TOP*, vol. 15, no. 2, pp. 161–198, 2007.

[17] R. Weber and G. Weiss, "On an index policy for restless bandits," *Journal of Applied Probability*, no. 27, pp. 637–648, 1990.

[18] K. Sevcik, "Scheduling for minimum total loss using service time distributions," *Journal of the ACM*, vol. 21, pp. 66–75, 1974.

[19] J. Gittins, *Multi-armed Bandit Allocation Indices*. Chichester: Wiley, 1989.

[20] W. Smith, "Various optimizers for single stage production," *Naval Res. Logist. Quart.*, vol. 3, pp. 59–66, 1956.

[21] D. Fife, "Scheduling with random arrivals and linear loss functions," *Management Science*, vol. 11, no. 3, pp. 429–437, 1965.

[22] C. Buyukkoc, P. Varaya, and J. Walrand, "The $c\mu$ rule revisited," *Adv. Appl. Prob.*, vol. 17, pp. 237–238, 1985.

[23] I. Meilijson and G. Weiss, "Multiple feedback at a single server station," *Stochastic Processes and Applications*, vol. 5, pp. 195–205, 1977.

[24] J. Gittins and D. Jones, "A dynamic allocation index for the sequential design of experiments," in *Progress in Statistics*, J. Gani, Ed. North-Holland, 1974, pp. 241–266.

[25] P. Whittle, "Arm-acquiring bandits," *Annals of Probability*, vol. 9, no. 2, pp. 284–292, 1981.

[26] P. Jacko, "Adaptive greedy rules for dynamic and stochastic resource capacity allocation problems," *Medium for Econometric Applications*, vol. 17, no. 4, pp. 10–16, 2009.

[27] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2005.