Resource-Sharing in a Single Server with Time-Varying Capacity

(Invited Paper)

Urtzi Ayesta*†, Martin Erausquin*‡, Peter Jacko*

*BCAM—Basque Center for Applied Mathematics, Bizkaia Technology Park Building 500, E-48160 Derio (Spain)
†IKERBASQUE—Basque Foundation for Science, Alameda Urquijo, 36-5, Plaza Bizkaia, E-48011 Bilbao (Spain)
‡UPV/EHU—Universidad del País Vasco, Avda. Lehendakari Agirre, 83, E-48015 Bilbao (Spain)

***Abstract*—We investigate the problem of sharing the resources of a single server with time-varying capacity with the objective of minimizing the mean delay. We formulate the resource allocation problem as a Markov Decision Process. The problem is not solvable analytically in full generality, and we thus set out to obtain an approximate solution. In our main contribution, we extend the framework of multi-armed bandits to develop a heuristic solution of index type. At every given time, the heuristic assigns an index to every user that depends solely on its current state, and serves the user with highest current index value. We show that in the case of constant capacity, the heuristic policy is equivalent to the so-called Gittins index rule, which is known to be optimal under the assumption of constant capacity.**

I. Introduction

Resource allocation and scheduling in queues is a classical field that have received a significant attention from the research community over the years. In the most studied problem, the capacity of the server is assumed constant, jobs arrive according to a random process, the service time distribution is general, and the objective is to minimize the expected mean delay (which by Little's law is equivalent to minimizing the mean number of jobs in the system) or, more generally, the expected total holding costs. Classical results show that the Shortest-Remaining-Processing-Time (SRPT) policy, minimizes the number of jobs sample-path wise [10], [11].

If the scheduling discipline does not have knowledge of the remaining service times of jobs, case usually referred to as "non-anticipating", Gittins [4], [5] proved that the so-called Gittins index rule minimizes the expected mean delay. The Gittins index calculates, based on the attained service of jobs, the optimum quantum of service that a job should obtain next. Other important related results show that if the service time distribution belongs to the Decreasing Hazard Rate (DHR) class, the Foreground-Background (FB) policy (a.k.a. Least-Attained-Service (LAS)), in which the job with the least attained service is always served, is optimal [16], [17], [8], whereas the ordinary First-Come-First-Served (FCFS) discipline, or any other non-preemptive discipline, minimizes the mean delay for the service time distributions belonging to the New Better than Used in Expectation (NBUE) class [9].

In the last decade technological developments in wireless systems and a growing concern for energy minimization have spurred the interest of investigating the problem of scheduling in systems where the available capacity might evolve over time. This is the case for instance in the speed-scaling problem where capacity is a controllable parameter [15] or in wireless systems where due to fading and interference effects, the quality of a downlink channel, and hence its transmission rate, fluctuates over time [12], [3], [2].

This paper represents a first attempt to investigate the problem of how to allocate the resources of a server among concurrent jobs, when the capacity of the server fluctuates over time. We consider a time-slotted system, and a finite initial population of jobs with a discrete service time, and we assume that the available capacity in each slot (including the future) is known to the scheduler. We wish to determine the "non-anticipating" discipline that minimizes the the *mean delay*. We formulate the problem in the framework of Markov decision processes as an extension of the multi-armed bandit problem [5]. We first show that the problem of minimizing the *mean delay*, where a user incurs a unit holding cost per slot while he is in the system, is equivalent to the problem of maximizing the *completion reward*, where a user receives a unit reward upon service completion. The latter is more amenable for analysis, and using Weber's interpretation of the index policy [13] we define a heuristic index rule for problem under consideration. We further show that in the constant capacity case, the heuristic policy is equivalent to Gittins' policy, which is known to be optimal in this case [5]. We provide several simple examples that show that the heuristic can be arbitrarily far from the optimal policy, which calls for substantial effort in the future.

The rest of the paper is organized as follows. In Section II we describe the problem. In Section III we formulate the Markov Decision Process model. In Section IV we develop a heuristic index rule. Section V presents several illustrating examples, and Section VI presents some concluding remarks.

This is the authors' version of the paper with the following reference: Ayesta, U., Erausquin, M., & Jacko, P. (2011): Resource-Sharing in a Single Server with Time-Varying Capacity, In Proceedings of Forty-Ninth Annual Allerton Conference on Communication, Control, and Computing, (IEEE Xplore to be confirmed). Emails: ayesta@bcamath.org, martin.erausquin@ehu.es, jacko@bcamath.org. Research partially supported by grant MTM2010-17405 (Ministerio de Ciencia e Innovación, Spain) and grant PI2010-2 (Department of Education and Research, Basque Government). Martin Erausquin's Ph.D. is supported by grants ECO2008-00777 and FPU AP2008-02014 (both of the MEC, Spain).

Proofs are deferred to the Appendix.

II. Problem Description

We consider a system operating in regular time slots, with one server sharing resources among a fixed set of users, $\mathcal{N} := \{1, \cdots, N\}$. Let us denote by X_i the discrete random variable that represents the service requirements of a given user $i \in \mathcal{N}$, defined in $\mathcal{X}_i := \{0, 1, \ldots, \text{maxsize}_i\}$. At the beginning of each time slot $t \in \mathcal{T} := \{0, 1, \cdots\}$, the server (scheduler) has to choose which user to serve. We assume that the capacity of the server varies over time. The server can give $c(t)$ units of service at time slot t, with $c(t) \geq 0$ (integer) for all t. The future evolution of the capacity function $c(t)$ is assumed to be known by the server at every time slot, i.e, the capacity function is deterministic and known by the server.

The probability distributions associated with X_i are known by the scheduler, for all $i \in \mathcal{N}$. However, at each time slot t, the scheduler does not have exact information about the remaining service requirements of the users. Instead, the server knows the amount of service that has been given (attained service) to each user $i \in \mathcal{N}$, which we denote by $x_i(t)$.

The server is allowed to be preemptive, i.e., it can stop giving resources to a user although its service requirements have not been completed.

A. Performance Criterion

The objective is to minimize the expected mean delay of the users in the system. We denote by Π the set of admissible policies which are preemptive, randomized, non-anticipating (with respect to remaining service requirements), and serve only one user in each time slot. Then, for a given policy $\pi \in \Pi$, T_i^π is the random variable denoting the number of time slots that user i spends in the system. The goal of the scheduler is to find a policy that solves the following optimization problem

$$\min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_0 \left[\sum_{t=0}^{T_i^\pi} \beta^t \right], \tag{1}$$

where $0 < \beta \leq 1$ is a given discount factor and $\mathbb{E}_t$ is the expectation conditional on the information known at the beginning of time slot t. Note that in the undiscounted case ($\beta = 1$), the problem is effectively that of minimizing the mean delay:

$$\min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_0 \left[T_i^\pi \right]. \tag{2}$$

III. MDP Model

The optimization problem described above can be formulated as a Markov decision processes (MDP). In the following we describe the elements of such a model.

A. Model of a User

Each user i is defined independently of other users as the tuple $\left(\mathcal{Z}_i, \mathcal{A}, (\boldsymbol{R}_i^a)_{a\in\mathcal{A}}, (\boldsymbol{P}_i^a)_{a\in\mathcal{A}}\right)$, where

- $\mathcal{Z}_i := \mathcal{X}_i \times \mathcal{L} \times \mathcal{T}$ is the three-dimensional *state space*, where $\mathcal{X}_i$ is the set of all possible levels of attained service, $\mathcal{L} := \{0, 1\}$ denotes whether the job is completed (0) or it is still in the system (1), and $\mathcal{T}$ is the slot-set defined above;
- $\mathcal{A} := \{0, 1\}$ the *action space*, where action 0 means *not to serve* user i, and action 1 means *to serve* the user;
- $\boldsymbol{R}_i^a := (R_i^a(z))_{z\in\mathcal{Z}_i}$, where $R_i^a(z)$ is the expected one-slot *reward* earned by user i at state z if action a is decided at the beginning of a slot;
- $\boldsymbol{P}_i^a := (p_i^a(n, m))_{n,m\in\mathcal{Z}_i}$ is the user-i stationary one-slot *state-transition probability matrix* if action a is decided at the beginning of a slot, i.e., $p_{i,n,m}^a$ is the probability of moving to state m from state n under action a.

The state of user i, $z_i = (x_i, l_i, t)$ thus consists of the attained service of the user (x_i), the information whether it has already left the system because their service requirements have already been satisfied (l_i), and the current time slot t which uniquely characterizes the subsequent capacity function $c(s), s \geq t$ of the system.

The one-slot state-transition probabilities are defined as follows:

$$\begin{aligned}
&p_i^1\left((x, 1, t), (x + c(t), 0, t + 1)\right) := \\
&\qquad \mathbb{P}\left[x < X_i \leq x + c(t) | X_i > x\right], \\
&p_i^1\left((x, 1, t), (x + c(t), 1, t + 1)\right) := \\
&\qquad 1 - \mathbb{P}\left[x < X_i \leq x + c(t) | X_i > x\right], \\
&p_i^0\left((x, 1, t), (x, 1, t + 1)\right) := 1, \\
&p_i^1\left((x, 0, t), (x, 0, t + 1)\right) := 1, \\
&p_i^0\left((x, 0, t), (x, 0, t + 1)\right) := 1.
\end{aligned}$$

The remaining probabilities are zero.

The dynamics of user i is thus captured by the *state process* $Z_i(\cdot)$ and the *action process* $a_i(\cdot)$, which correspond to state $Z_i(t) \in \mathcal{Z}_i$ and action $a_i(t) \in \mathcal{A}$ at all time slots $t \in \mathcal{T}$. As a result of deciding action $a_i(t)$ in state $Z_i(t)$ at time slot t, the user i consumes the allocated capacity, earns the reward, and evolves its state for the time slot $t + 1$ (see Subsection III-C for more details).

B. Model of the System

The state of the system (*joint state*) is characterized by the attained service of each user, the information about the users that have already left the system because their service requirements have been satisfied, and by the capacity of the system from that moment on. Thus, the state of the system at time slot t is $(\vec{x}(t), \vec{l}(t), t)$, where $\vec{x}(t) = (x_1(t), \cdots, x_N(t))$, and $\vec{l}(t) = (l_1(t), \cdots, l_N(t))$.

In time slot t, the server knows the joint state $(\vec{x}(t), \vec{l}(t), t)$ and it chooses an action $a_i(t)$ for all the users, with $a_i(t) \in \mathcal{A}$. Since only one user can be served within one time slot, the action is chosen with the following constraint: $\sum_{i=1}^{N} a_i(t) =$

$1\ \forall t$. Depending on the action chosen, the server earns the corresponding rewards from all the users at the end of the time slot.

Then, the optimization problem associated with such a Markov decision process can be written as:

$$\max_{\pi\in\Pi}\mathbb{E}_0\left[\sum_{t=0}^{\infty}\sum_{i=1}^{N}\beta^t R_i^{a_i(t)}\left(x_i(t),l_i(t),t\right)\right], \qquad (3)$$

$$\text{subject to } \sum_{i=1}^{N} a_i(t)=1,\ \text{for all } t\in\mathcal{T}.$$

Note that there are two features that make the above problem extremely challenging. First, the sample path allocation constraint is well-known for causing intractability in similar problems, such as the weakly-coupled MDPs [6] or restless bandits [14]. Second, the interdependence of the user states (by the dependence on the common factor t) is a feature for which few results exist. As a consequence of these two features, the objective shifts to finding optimal solutions for special cases, or designing approximate general solutions.

C. Reward Definition

In this paper we will consider two reward structures. We will talk about the *completion reward problem*, if the expected one-slot reward is defined as the expected number of completed jobs. Thus reward is given by the completion probability, that is,

$$R_i^1(x,1,t):=\mathbb{P}\left[x<X_i\le x+c(t)|X_i>x\right],$$
$$R_i^0(x,1,t):=0, \qquad R_i^a(x,0,t):=0.$$

On the other hand, we will talk about the *mean delay problem*, if the expected one-period reward is defined as the expected unitary cost of remaining in the system, i.e., as follows:

$$R_i^1(x,1,t):=-(1-\mathbb{P}\left[x<X_i\le x+c(t)|X_i>x\right]),$$
$$R_i^0(x,1,t):=-1, \qquad R_i^a(x,0,t):=0.$$

Note that, if $0<\beta<1$ and if $\beta=1$, then the mean delay problem defined with the latter reward structure, is the same as (1) and (2), respectively, multiplied by N. Next we establish a relationship between the completion reward problem and the mean delay problem, on which we will rely in the following sections.

Proposition 1: Suppose that $0<\beta<1$ is given. Then, policy π_1 achieves higher objective value (i.e., performs better) in the completion reward problem than policy π_2 if and only if policy π_1 achieves higher objective value (i.e., performs better) in the mean delay problem than policy π_2. As a consequence, π is optimal for the completion reward problem if and only if it is optimal for the mean delay problem.

This result means that for every $0<\beta<1$ the completion reward problem and the mean delay problem are essentially equivalent. Notice, however, that the completion reward problem in the undiscounted case ($\beta=1$) is optimally solved by any non-idling policy, yielding the objective value equal N. On the other hand, this is not the case with the mean delay problem, in which different non-idling policies in general, perform differently. However, we can obtain an optimal policy to this problem in the limit.

Proposition 2: The optimal solution to the undiscounted mean delay problem given in (2) is obtained by taking the limit $\beta\to 1$ of the optimal solution to the discounted completion reward problem.

The above two results justify our interest in studying the completion reward problem in both the discounted and undiscounted version. In Section IV we will design a novel index rule that we propose to (approximately) solve (1) and (2).

D. Exact Solution

From dynamic programming it follows that the value function of the problem (3) in terms of the completion reward problem is the unique solution of the Bellman equation,

$$\begin{aligned}\Phi(\vec{x},\vec{l},t)=\max\Big\{&\beta\Phi(\vec{x},\vec{l},t+1);\max_{i\text{ s.t. }l_i=1}\big\{R_i^1(x_i,l_i,t)+\\&+\beta R_i^1(x_i,l_i,t)\,\Phi(\vec{x}+c(t)\cdot\vec{e}_i,\vec{l}-e_i,t+1)\\&+\ \beta(1-R_i^1(x_i,l_i,t))\Phi(\vec{x}+c(t)\cdot\vec{e}_i,\vec{l},t+1)\big\}\Big\}\end{aligned} \qquad (4)$$

where $\vec{e}_i$ represents an N-dimensional vector with all components equal to 0, except for the i^{th} component, which is equal to 1. Note that the first term, $\beta\Phi(\vec{x},\vec{l},t+1)$ refers to idling (or serving any of the users that have already left).

Characterizing the optimal solution as well as its numerical computation is not feasible in general because of the combinatorial explosion and huge, $(2N+1)$-dimensional state space. We will therefore focus on proposing an approximate solution next.

IV. An Index Rule

In this section we propose a novel index rule for this model, where an index rule is defined as follows: *At the beginning of each time slot, an index value is calculated for any user in the system, and the scheduler serves the user with the highest index value within the following time slot.*

For a given user $i\in\mathcal{N}$ with uncompleted service, this index value depends only on its own state, which captures the attained service up to the time slot where the decision is taken, and the capacity of the system from that moment on. For every job $i\in\mathcal{N}$ in its state $(x_i,1,t)$ we define the following index value:

$$K_i(x_i,1,t):=\sup_{\tau\ge 1}\frac{\sum_{k=0}^{\tau-1}\beta^k\mathbb{P}\left[x_i+s_{t,k}<X_i\le x_i+s_{t,k+1}\right]}{\sum_{k=0}^{\tau-1}\beta^k\mathbb{P}[X_i>x_i+s_{t,k}]} \qquad (5)$$

where

$$s_{t,k}:=\sum_{l=0}^{k-1}c(t+l).$$

Note that in the undiscounted case ($\beta = 1$), the above formula simplifies to

$$K_i(x_i, 1, t) = \sup_{\tau \geq 1} \frac{\mathbb{P}\left[x_i < X_i \leq x_i + s_{t,\tau}\right]}{\sum_{k=0}^{\tau-1} \mathbb{P}[X_i > x_i + s_{t,k}]} \tag{6}$$

We set $K_i(x_i, 0, t) := 0$, for all the jobs $i \in \mathcal{N}$ that have already left the system, in order to ensure that an uncompleted job is always given priority over a job that has already left the system. The index rule consists of giving service, at each time slot t, to the user with the highest index value. Formally, the user i is served at time slot t if

$$i \in \text{argmax}_j\, K_j(x_j(t), l_j(t), t). \tag{7}$$

In Proposition 3 we present the justification and derivation of the index value given by (6), which is based on the formulation presented by Weber [13] to design and prove optimality of the Gittins index for the classical multi-armed bandit problem. But first we provide an intuitive explanation. First, note that any scheduling policy can be seen as making two different decisions. It decides which user to serve, and it also decides when to stop serving this user. The number of slots until the first time slot at which another job is considered to being served is called the *stopping time*, which we denote by $\tau \in \mathcal{T}$.

Consider the completion reward problem with only one user, labeled as i, and assume moreover that the system has to pay a charge M_i for every slot whenever the server serves this user. Suppose that the service must be provided continuously until a stopping time, after which the user is not allowed to be served anymore. Since this user is the only one in the system, the server has to decide whether or not to serve it. On the other hand, it also needs to specify the stopping time, i.e., the time slot in which the processing of the job will be stopped.

At any time slot $t \in \mathcal{T}$ in which the job is uncompleted, and for any stopping time $\tau \geq 1$, there is a value of the charge M_i, such that the expected cost to the scheduler is exactly in balance with the expected rewards to be obtained by processing the job until τ. Such a value of the charge is called the *fair charge*, and it depends on the state of the job at time slot t, i.e., both on t and $x_i(t)$. Thus, the fair charge $M_i(x_i(t), t, \tau)$, is the value of M_i that solves the following equation:

$$\mathbb{E}_t \left[\sum_{k=0}^{\tau-1} \beta^k \left(R_i^1\left(x_i(t+k), l_i(t+k), t+k\right)\right.\right.$$
$$\left.\left. - M_i 1_{\{l_i(t+k)=1\}}\right)\right] = 0. \tag{8}$$

Note that, although the stopping time is fixed, the cost for the scheduler is a random variable, since the job can leave the system before the stopping time is reached.

We denote by $M_i(x_i(t), t)$ the maximum fixed charge which accepts a stopping time for which this value is its corresponding fair charge, i.e.,

$$M_i(x_i(t), t) = \sup_{\tau \geq 1} M_i(x_i(t), t, \tau).$$

In the next proposition we show that, in fact, the fair charge coincides with the index value proposed in (6).

Proposition 3: For all t and for all $x_i(t)$ we have $M_i(x_i(t), t) = K_i(x_i(t), 1, t)$.

In view of (13) the index value can be written as

$$K_i(x_i, 1, t) = \sup_{\tau \geq 1} \frac{\mathbb{R}_i^\tau(x_i(t), 1, t)}{\mathbb{W}_i^\tau(x_i(t), 1, t)}.$$

where $\mathbb{R}_i^\tau(x_i(t), 1, t)$ represents the expected total (discounted) reward earned by user i if it were served for τ time slots, and $\mathbb{W}_i^\tau(x_i(t), 1, t)$ the expected total (discounted) time spent in the system by user i if the server decided to give service to this user for τ time slots.

A. Optimality of the Index Rule with constant capacity

In the case of constant capacity, $c(t) = 1, \forall t \in \mathcal{T}$, the so-called Gittins index rule is known to be optimal [5]. This policy consists, first, of calculating the following *Gittins index* value for each (uncompleted) user:

$$J_i(x_i) = \sup_{\tau \geq 1} \frac{\sum_{k=0}^{\tau-1} \beta^k \mathbb{P}\left[x_i + k < X_i \leq x_i + (k+1)\right]}{\sum_{k=0}^{\tau-1} \beta^k \mathbb{P}[X_i \geq x_i + (k+1)]}. \tag{9}$$

and then, processing the job i^* with the highest index value:

$$i^* = \text{argmax}_{i: l_i = 1}\, J_i(x_i) \tag{10}$$

Thus, the scheduler serves the user with the highest index value within one time slot, and at the end of this time slot, it calculates again the indices under the new state of the system.

From (6) and (7) it follows that in the case of constant capacity our index rule reduces to Gittins index policy.

The Gittins index of a given user $i \in \mathcal{N}$ depends only on the attained service, and not on time. This means that the index value remains constant if the job does not receive any service. In [1] the authors prove the following result: if $j = \text{argmax}_i\, J_i(x_i)$, and $\tau^* = \text{argmax}_\tau \frac{\mathbb{R}_j^\tau(x_j)}{\mathbb{W}_j^\tau(x_j)}$, then, the index value of user j is never below $J_i(x_i)$, at least until τ^* is reached. Therefore, the optimal policy is to serve user j for at least τ^* time slots. Moreover, the index value of a user which is not served does not change, since it only depends on attained service. Since the index of each user depends only on its own attained service, it is easy to calculate, and therefore, the optimal policy can be implemented in an efficient way.

The fact that the index value remains constant if a job is not served is crucial when proving the optimality of Gittins index policy. However, this property does not hold in the setting with time-varying capacity: the proposed index value of a job can change from one time slot to other, even in the case where this job has not received any service from the system, since it depends in t and in turn on the future capacity.

On the other hand, one could propose an index which is independent of t. However, then the proof of optimality of the Gittins index rule does not extend neither, since an index independent of t can not correctly capture the future evolution of the system.

V. Examples

We have analyzed the performance of our heuristic defined by the index rule (7) in several numerical experiments, and compared it with both the Gittins index policy and the optimal policy. We have considered systems with only 2 users, changing the capacity function and the service requirement distributions from one scenario to another. The optimal policy has been obtained using the value iteration algorithm.

Our numerical experiments (not reported here), suggest the following results:

- In some cases, our heuristic is optimal, whereas Gittins index policy is not.
- Our heuristic rule is always at least as good as Gittins index policy.
- Both our heuristic and the Gittins index policy can be arbitrarily far from the optimal policy.

In the rest of this section, we present two of those scenarios. The first scenario gives an intuition as to why Gittins index is not optimal with time varying capacity, and illustrates how our heuristic captures crucial information that allows it to be optimal. In the second scenario, we show that our heuristic can be suboptimal as well, and we observe that both Gittins index policy and our heuristic can incur in an arbitrarily large suboptimality gap.

In both scenarios, we consider only two users in the system, we assume that the attained service of both users is $x_i(0) = 0$ at $t = 0$, and we fix $\beta = 1$.

A. Scenario I

Suppose that the distribution of the service requirements of the two users follow these random variables:

$$X_1 = 11, \text{ with probability } p_1 = 1.$$

$$X_2 = \begin{cases} 9 & \text{with probability } p_2 = 0.1, \\ 21 & \text{otherwise.} \end{cases}$$

Consider that the capacity of the system is given by the following function:

$$c(t) = \begin{cases} 1 & \text{if } 0 \le t \le 9, \\ 0 & \text{if } 10 \le t \le 49, \\ 5 & \text{if } 50 \le t. \end{cases}$$

Thus, the system has capacity 1 for the first 10 time slots. Then, for 40 time slots, the system can not serve any user. At $t = 50$, the system starts giving service again, but in this case, it can give 5 units of service within each time slot.

The Gittins index value at time $t = 0$ for job 1 is $\frac{1}{11} = 0.090$, while the Gittins index value for job 2 is 0.0529. Moreover, the index value of job 1 is bigger than that of job 2 in all the time slots, until job 1 is completed. Thus, the Gittins index consists of starting serving user 1 until its completion, and then serving user 2. Under this policy, labeled by $\pi_{1,2}$ the completion time for job 1 will be $T_1^{\pi_{1,2}} = 51$ with probability 1. After T_1, the system will start giving service to user 2. With probability 0.1, the completion time will be $T_2^{\pi_{1,2}} = 53$, and otherwise, $T_2^{1,2} = 56$. If we measure the performance of this policy in terms of the mean delay problem we observe that the expected mean delay is given by:

$$\frac{1}{2}\sum_{i=1}^{2} \mathbb{E}\left[T_i^{\pi_{1,2}}\right] = \frac{51 + (0.1 \cdot 53 + 0.9 \cdot 56)}{2} = 53.35.$$

It is clear that, following this scheduling rule, both users will be in the system for at least 51 time units. The policy does not take any advantage of the positive probability of completing job 2 before the system's capacity becomes 0 at $t = 10$.

Consider the policy, labeled as $\pi^{2,1}$, which starts giving service to user 2 until its completion, and then serves user 1. From (6) it readily follows that our heuristic is in fact equivalent to $\pi_{2,1}$. For instance, index value at $t = 0$ for job 2 is 0.0210, while the index value for job 1 is 0.02. In this case, the completion times for both users are given by:

$$T_2^{\pi_{2,1}} = \begin{cases} 9 & \text{with probability } p_2 = 0.1, \\ 53 & \text{otherwise.} \end{cases}$$

$$T_1^{\pi_{2,1}} = 56 \text{ with probability } p_1 = 1,$$

and therefore, the measure of the performance of the system

$$\frac{1}{2}\sum_{i=1}^{2} \mathbb{E}\left[T_i^{\pi_{2,1}}\right] = \frac{56 + (0.1 \cdot 9 + 0.9 \cdot 53)}{2} = 52.30.$$

In other words, our heuristic outperforms the Gittins index rule.

Moreover, the value iteration algorithm has shown that the policy $\pi_{2,1}$, and hence our heuristic as well, is in fact optimal.

B. Scenario II

The following example, with a binary capacity, illustrates that our heuristic needs not be optimal in general. We define the random variables that represent the service requirements of the two users as:

$$X_1 = 5, \text{ with probability } p_1 = 1.$$

$$X_2 = \begin{cases} 1 & \text{with probability } p_2 = 0.3, \\ 6 & \text{otherwise.} \end{cases}$$

The capacity of the system is given by the function:

$$c(t) = \begin{cases} 1 & \text{if } 0 \le t \le 4, \\ 0 & \text{if } 5 \le t \le 9, \\ 1 & \text{if } 10 \le t. \end{cases}$$

Thus, the system has capacity 1 for 5 time slots, then it is OFF for other 5 slots, and then it starts giving service again, with capacity 1.

It is not clear without further calculations, which is the optimal policy. The server could be tempted to start serving user 2, since it could leave the system after only 1 time slot, with probability 0.3. On the other hand, if the server starts serving user 1, and keeps serving it for the first 5 time slots, this user will leave the system with probability 1 before the system goes OFF.

It can be shown that the optimal policy starts giving service to user 1 until its completion, and then serves user 2. Following this policy, labeled by $\pi_{1,2}$, the system loses the opportunity of completing user 2's service requirements in only 1 time slot, but it ensures that user 1 will leave the system before the time period where the system is OFF. The completion times are the following:

$$T_1^{\pi_{1,2}} = 5.$$

$$T_2^{\pi_{1,2}} = \begin{cases} 11 & \text{with probability } p_2 = 0.3, \\ 16 & \text{otherwise.} \end{cases}$$

$$\frac{1}{2}\sum_{i=1}^{2} \mathbb{E}\left[T_i^{\pi_{1,2}}\right] = \frac{5 + (0.3 \cdot 11 + 0.7 \cdot 16)}{2} = 9.75.$$

In this example, our heuristic and Gittins index policy are equivalent. They first start giving service to user 2. User 1 will be in the system for at least 11 time slots, and the probability of having both users queueing for at least 11 time slots is strictly positive. After one time slot, both scheduling policies are indifferent between serving user 1 or 2, since both jobs need 5 units of capacity until their completion. Consider, for instance, the performance of these two policies if they start serving user 2 at $t = 0$ until its completion, and then serve user 1, and label by $\pi_{2,1}$ this policy. Then, the completion times are given by:

$$T_1^{\pi_{2,1}} = \begin{cases} 11 & \text{with probability } p_2 = 0.3 \\ 16 & \text{otherwise} \end{cases}$$

$$T_1^{\pi_{2,1}} = \begin{cases} 1 & \text{with probability } p_2 = 0.3 \\ 11 & \text{otherwise} \end{cases}$$

$$\begin{aligned} \frac{1}{2}\sum_{i=1}^{2} \mathbb{E}\left[T_i^{\pi_{2,1}}\right] &= \frac{(0.3 \cdot 11 + 0.7 \cdot 16) + (0.3 \cdot 1 + 0.7 \cdot 11)}{2} \\ &= 11.25. \end{aligned}$$

Thus, in this case it is better to serve user 1 within the first 5 time slots, ensuring that it will leave the system after this interval. This example shows that our heuristic is not always optimal in the time-varying setting.

Note also that the suboptimality gap of the index policies increases as the number of time slots when the system is OFF gets larger.

VI. Conclusions

Scheduling in systems with a time-varying capacity is an important problem with applications in various contexts. In full generality, the problem can not be solved analytically, which motivates us to resort to seek an approximate solution. In our main contribution we use the framework of multi-armed bandits to derive a simple heuristic index rule. We present two examples to illustrate that even though our index rule consistently outperforms Gittins' index rule, it can nevertheless incur a large sub-optimality gap. Thus, more research is needed in order to characterize under which conditions our index rule provides a satisfactory performance. In future work we plan to extend the framework to accommodate for a stochastic evolution of the capacity in the system, and general service time distributions.

VII. Acknowledgements

The authors are thankful to John Gittins (University of Oxford) for interesting discussions, and to Rhonda Righter (UC Berkeley) for discussions and for providing the idea of the example in Scenario I.

References

[1] S. Aalto, U. Ayesta, and R. Righter. On the optimal scheduling discipline in a single-server queue. *Queueing Systems (special issue The Erlang Centennial)*, 63(1–4):437–458, 2009.

[2] U. Ayesta, M. Erausquin, and P. Jacko. A modeling framework for optimizing the flow-level scheduling with time-varying channels. *Performance Evaluation*, 67:1014–1029, 2010.

[3] S.C. Borst. User-level performance of channel-aware scheduling algorithms in wireless data networks. *IEEE/ACM Transactions on Networking*, 13(3):636–647, 2005.

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

[5] J.C. Gittins, K. Glazebrook, and R. Weber. *Multi-armed Bandit Allocation Indices*. Wiley, 2011.

[6] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 165–172, 1998.

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

[8] R. Righter and J.G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Probability in the Engineering and Informational Sciences*, 3:323–334, 1989.

[9] R. Righter, J.G. Shanthikumar, and G. Yamazaki. On extremal service disciplines in single-stage queueing systems. *Journal of Applied Probability*, 27:409–416, 1990.

[10] L.E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690, 1968.

[11] D.R. Smith. A new proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 26:197–199, 1978.

[12] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, 39(2):466–478, 1993.

[13] R. Weber. On the Gittins index for multiarmed bandits. *Annals of Applied Probability*, 2(4):1024–1033, 1992.

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

[15] A. Wierman, L.L.H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *Proceedings of IEEE INFOCOM*, 2009.

[16] S.F. Yashkov. On feedback sharing a processor among jobs with minimal serviced length (in russian). *Technika Sredstv Svyazi. Ser. ASU*, 2:51–62, 1978.

[17] S.F. Yashkov. Processor sharing queues: Some progress in analysis. *Queueing Systems*, 2:1–17, 1987.

Appendix

Proof of Proposition 1

Proof: First, let us reformulate the problems with the two reward schemes defined in Section III. The completion reward problem is

$$\max_{\pi \in \Pi} \sum_{i=1}^{N} \mathbb{E}_0 \left[\beta^{T_i^\pi}\right], \qquad (11)$$

while the mean delay problem can be written as

$$\min_{\pi\in\Pi}\sum_{i=1}^{N}\mathbb{E}_0\left[1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1}\right]. \tag{12}$$

Consider a fixed policy π and a fixed user $i\in\mathcal{N}$. Then, it is known that

$$1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1}=\frac{1-\beta^{T_i^\pi}}{1-\beta}.$$

Therefore, we have

$$1-(1-\beta)(1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1})=\beta^{T_i^\pi}.$$

If we sum these values obtained over all the users, we obtain

$$N-(1-\beta)\sum_{i=1}^{N}(1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1})=\sum_{i=1}^{N}\beta^{T_i^\pi}.$$

Taking expectations in both sides of the equation, we obtain that

$$N-(1-\beta)\sum_{i=1}^{N}\mathbb{E}_0\left[1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1}\right]$$
$$=\sum_{i=1}^{N}\mathbb{E}_0\left[\beta^{T_i^\pi}\right].$$

Note that the sums on the left-hand side and on the right-hand side are the objective values under policy π for the mean delay problem and for the completion reward problem, respectively. Since N and $(1-\beta)$ are strictly positive numbers, then the relative order of performance of policies π_1 and π_2 is the same for both problems. As a consequence, also the optimal policies are the same, i.e.,

$$\mathrm{argmin}_{\pi\in\Pi}\sum_{i=1}^{N}\mathbb{E}\left(1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1}\right)$$
$$=\mathrm{argmax}_{\pi\in\Pi}\sum_{i=1}^{N}\mathbb{E}\left(\beta^{T_i^\pi}\right).$$

which finishes the proof. ∎

Proof of Proposition 2

Proof: By Proposition 1 we have that the optimal solution to the discounted completion reward problem is also an optimal solution to the discounted mean delay problem. Then, invoking [7, Lemma 7.1.8] we get that the optimal policy for the undiscounted mean delay problem is obtained by the limit $\beta\to1$ of the optimal policy for its discounted version, since it is clear that the limit of the discounted problem

$$\min_{\pi\in\Pi}\lim_{\beta\to1}\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_0\left[1+\beta+\beta^2+\cdots+\beta^{T_i^\pi-1}\right],$$

is equal to the undiscounted problem (2). ∎

Proof of Proposition 3

Proof: From (8), we obtain, for every τ, that

$$M_i(x_i(t),t,\tau)=\frac{\mathbb{E}_t\left[\sum_{k=0}^{\tau-1}\beta^kR_i^1\left(x_i(t+k),l_i(t+k),t+k\right)\right]}{\mathbb{E}_t\left[\sum_{k=0}^{\tau-1}\beta^k(1_{\{l_i(t+k)=1\}})\right]}. \tag{13}$$

Substituting the definitions given for $R_i^1\left(x_i(t+k),1,t+k\right)$ in Section III-C, and applying the definition of conditional expectation, we have that the right-hand side expression is equal to

$$\frac{1}{\mathbb{E}_t\left[\sum_{k=0}^{\tau-1}\beta^k(1_{\{l_i(t+k)=1\}})\right]}\times$$
$$\times\left[\sum_{k=0}^{\tau-1}\beta^k\mathbb{P}\left(l_i(t+k)=1|X_i>x_i(t)\right)\cdot\right.$$
$$\left.\cdot\mathbb{P}\left[x_i(t)+s_{t,k}<X_i\le x_i(t)+s_{t,k+1}|X_i>x_i(t)+s_{t,k}\right]\right]. \tag{14}$$

From the definition of conditional expectation, we have, for every value of k,

$$\mathbb{P}\left(l_i(t+k)=1|X_i>x_i(t)\right)=\frac{\mathbb{P}\left(X_i>x_i(t)+s_{t,k}\right)}{\mathbb{P}\left(X_i>x_i(t)\right)}$$

and

$$\mathbb{P}\left[x_i(t)+s_{t,k}<X_i\le x_i(t)+s_{t,k+1}|X_i>x_i(t)+s_{t,k}\right]$$
$$=\frac{\mathbb{P}\left[x_i(t)+s_{t,k}<X_i\le x_i(t)+s_{t,k+1}\right]}{\mathbb{P}\left[X_i>x_i(t)+s_{t,k}\right]},$$

and therefore (14) can be simplified, obtaining

$$\frac{\sum_{k=0}^{\tau-1}\beta^k\mathbb{P}\left[x_i(t)+s_{t,k}<X_i\le x_i(t)+s_{t,k+1}\right]}{\mathbb{P}\left(X_i>x_i(t)\right)\mathbb{E}_t\left[\sum_{k=0}^{\tau-1}\beta^k(1_{\{l_i(t+k)=1\}})\right]}. \tag{15}$$

On the other hand, for the term in the denominator we have, for every k,

$$\mathbb{E}_t\left[(1_{\{l_i(t+k)=1\}})\right]=\frac{\mathbb{P}\left(X_i>x_i(t)+s_{t,k}\right)}{\mathbb{P}\left(X_i>x_i(t)\right)}.$$

Thus, from (15), we obtain

$$\frac{\sum_{k=0}^{\tau-1}\beta^k\mathbb{P}\left[x_i(t)+s_{t,k}<X_i\le x_i(t)+s_{t,k+1}\right]}{\sum_{k=0}^{\tau-1}\beta^k\mathbb{P}[X_i>x_i(t)+s_{t,k}]},$$

which is the same as (5), and therefore, we conclude that

$$\sup_{\tau\ge1}M_i(x_i(t),t,\tau)=M_i(x_i(t),t)=K_i(x_i(t),1,t),$$

which is the desired result. ∎