Equilibria of Online Scheduling Algorithms

Itai Ashalgi
Sloan School of Management, MIT
iashlagi@mit.edu

Brendan Lucier
Microsoft Research New England
brlucier@microsoft.com

Moshe Tennenholtz
Microsoft Research, Herzlyia, Israel
moshet@ie.technion.ac.il

Abstract

We describe a model for competitive online scheduling algorithms. Two servers, each with a single observable queue, compete for customers. Upon arrival, each customer strategically chooses the queue with minimal expected wait time. Each scheduler wishes to maximize its number of customers, and can strategically select which scheduling algorithm, such as First-Come-First-Served (FCFS), to use for its queue. This induces a game played by the servers and the customers.

We consider a non-Bayesian setting, where servers and customers play to maximize worst-case payoffs. We show that there is a unique subgame perfect safety-level equilibrium and we describe the associated scheduling algorithm (which is not FCFS). The uniqueness result holds for both randomized and deterministic algorithms, with a different equilibrium algorithm in each case.

When the goal of the servers is to minimize competitive ratio, we prove that it is an equilibrium for each server to apply FCFS: each server obtains the optimal competitive ratio of 2.

1 Introduction

Service providers often compete over customers. In this paper we study the effect of waiting time on competition. For example, a customer choosing between restaurants of similar price and quality might select the one with fewer waiting patrons. If competing cloud service providers have similar pricing structures, a client with a job to run might choose the provider with fewer jobs in queue. This customer behavior incentivizes servers to strategically manage their queues.

We consider a model in which two servers, each holding a queue, compete over self-interested customers who choose servers strategically to minimize wait time. We assume that the queues are *observable* to both customers and servers. Moreover, each server can apply an arbitrary *scheduling algorithm* that determines how customers are selected from its queue. The competition between servers defines a game: each server chooses a scheduling algorithm with the goal of maximizing the number of customers they will serve, and each customer strategically chooses a queue (upon arrival) to minimize wait time, given the selected protocols. We ask: what scheduling algorithms arise as equilibria of this game, and how will the market split as a result?

Each scheduling protocol can be viewed as an online algorithm, making service decisions based on an observed history. Following the theory of online algorithms, we adopt a non-Bayesian approach, meaning that there are no probabilistic assumptions on arrival and service rates. This yields an online game, which we analyze using two different non-Bayesian solution concepts. We first study *safety-level equilibria*, which are strategy profiles in which each player maximizes her worst-case utility, over all possible inputs, given the strategies of the other players (Ashlagi, Monderer, and Tennenholtz 2009; 2006). As the game is played sequentially we focus on *subgame perfect* equilibria: the choices of the customers as well as the online decisions of the algorithms must be rational at every potential state of the queues.

We also study *competitive ratio equilibria*: strategy profiles in which each player maximizes competitive ratio (i.e. worst-case approximation to the optimal utility in hindsight) given the others' strategies (Engelberg and Naor 2007). Here, too, we are interested in subgame perfect equilibria.

To provide insight into issues raised by our game-theoretic scenario, consider the First Come First Served (FCFS) scheduling algorithm. While natural, FCFS is not always a best response in our game. Suppose a customer c arrives and observes that one queue has 5 customers, and the other has 1. The second server could attract c by placing him next in line, as in FCFS. But the server can do better: c would join even if the server reserved the right to serve up to three more customers ahead of c. This is strictly better for the server, as it gives added flexibility for future customers.

This insight can be used to develop a strategy (i.e. a novel algorithm) called the *Promise algorithm*. This protocol offers each incoming customer the longest possible wait-time that beats the opponent's, if possible. The difficult part of this algorithm is correctly guessing the opponent's wait time, which must be inferred from the history of customer choices.

We find that, if servers are restricted to use deterministic algorithms, the profile in which both servers adopt the Promise algorithm is the *unique* subgame perfect safety-level equilibrium. This uniqueness result covers non-symmetric and mixed strategy profiles. We also show how to extend the promise algorithm to a randomized setting, where it offers an *expected* waiting time to each arriving customer. We again prove that this *Probabilistic Promise Algorithm* is the *unique* subgame perfect safety-level equilibrium. Prov-

ing uniquness in the randomized setting presents additional challenges, as there are many ways that an expected waiting-time can be realized; nevertheless we prove that there is a uniquely dominant realization method, which we describe.

When the servers' goal is to minimize competitive ratio, we show it is an equilibrium for each server to use FCFS. That is, FCFS optimizes competitive ratio when played against itself, and this optimal competitive ratio is 2. By comparison, the competitive ratio of Promise versus itself is strictly worse (at least 2.5), and hence FCFS is superior for the decision criterion of competitive ratio.

Related work As far as we are aware, our work is the first work to study competing queues with incomplete information over arrivals when *both servers and customers are strategic*. The closest work to ours is by (Hassin 2009), who studies customer equilibria when one scheduler uses FCFS and the other chooses customers randomly. In that work only the customers are strategic, and the arrival/service processes are stochastic. Competition between schedulers in a complete information setting was studied by (Ashlagi, Tennenholtz, and Zohar 2010). (Immorlica et al. 2011) studied a family of zero-sum games for two players, each of whom optimizes online to outperform the opponent. Our model can be viewed as an online zero-sum game, but servers accumulate utility rather than maximizing the probability of "winning" (another example is the competition on hiring a secretary (Immorlica, Kleinberg, and Mahdian 2006)). Moreover, only the algorithm designers are strategic in their model. A related topic is competition between sellers who seek to maximize revenue by attracting more customers (see, e.g., (McAfee 1993; Peters and Severinov 1997; Burguet and Sakovics 1999)).

Suboptimality of FCFS has been observed in other contexts. For example, Last Come First Served (LCFS) can be better when customers observe the queue before deciding whether to join ((Naor 1969), (Hassin 1985; Alperstein 1988)). These observations are for the single-server case.

2 Model and Preliminaries

Our setting is an online game played by two servers, $\{1, 2\}$, and a set C of n customers[1]. The game unfolds as a sequence of time-ordered events. Formally, an *arrival event*, (t, c), indicates customer $c \in C$ arrives at time t. A *queue choice event*, (t, c, j), indicates $c \in C$ joins queue j. A *service event*, (t, j), indicates server j finishes a service at t.

A *history* H is a sequence of time-ordered events. We say H' extends H, $H' \succeq H$, if H is a prefix of H'. A history H is *valid* if every $c \in C$ has at most one arrival and one queue choice, with any queue choice by c immediately following c's arrival, and every $H' \preceq H$ has at least as many queue choices for j as services by j. Write $\mathcal{H}$ for the set of valid histories. Write $C_j(H) = \{c \in C \colon \exists t \text{ s.t. } (t, c, j) \in H\}$, the customers who join queue $j \in \{1, 2\}$ throughout $H \in \mathcal{H}$.

A strategy for server j is a function $\mathcal{A} \colon \mathcal{H} \to C \cup \{nil\}$. We interpret $\mathcal{A}(H)$ as the customer to serve next if a service completion occurs following H. Given H and service event (t, j) following H, we say c *begins service* at time t if $\mathcal{A}(H) = c$. Write $Q_j(\mathcal{A}, H)$ for those customers who are "in queue" after H (i.e. joined j but haven't begun service), given j uses strategy $\mathcal{A}$. We require that $\mathcal{A}(H) \in Q_j(\mathcal{A}, H)$ when $Q_j(\mathcal{A}, H) \neq \emptyset$, or nil otherwise (that is, a server never remains idle while there are customers in queue). For example, the First-Come-First-Served (FCFS) scheduler, $\mathcal{F}$, selects the customer from $Q_j(\mathcal{F}, H)$ who arrived earliest (or nil if $Q_j(\mathcal{F}, H) = \emptyset$). A strategy for a customer $c \in C$ is a function $\sigma \colon \mathcal{H} \to \{1, 2\}$. We interpret $\sigma(H)$ as the customer's choice of server if he arrives following the events[2] in H. We write $\boldsymbol{\sigma}$ for a profile of strategies for each $c \in C$, and $\boldsymbol{\sigma}_{-c}$ for a profile that excludes customer c.

Arrival and service times are exogenous and initially unknown to the players. Formally, each $c \in C$ has an *arrival time* a_c, and each server j has an infinite sequence of *service times* $d_j^1 < d_j^2 < \dots$. Write Π for a set of arrival and service times, which we view as an input instance. Given input profile Π and strategies $\boldsymbol{\sigma}$ and $(\mathcal{A}_1, \mathcal{A}_2)$, the game resolves as follows. Write H_t for the set of events that occur strictly before time $t \geq 0$, with $H_0 = \emptyset$. For each $c \in C$, there is an arrival event (a_c, c) and choice event $(a_c, c, \sigma_c(H_{a_c})$. Whenever a customer joins an empty queue (i.e. $\sigma_c(H_t) = j$ and $Q_j(\mathcal{A}_j, H_t) = \emptyset$ where $t = a_c$) or there is a service event (t, j) with $\mathcal{A}_j(H_t) = c$, we say customer c *begins service from j at time t*; write $b(c)$ for this time t. Customer c *finishes being served* at time $f(c) = \min\{d_j^k \colon d_j^k > b(c)\}$; that is, at the next service time for j that follows $b(c)$. For each $c \in C$ there is a service event $(f(c), j)$ indicating the completion of service for customer c at time $f(c)$. As there are $3n$ events and all a_c, d_j^k are finite, there is some $T \geq 0$ such that no event occurs after time T. Write $H(\Pi) = H(\mathcal{A}_1, \mathcal{A}_2, \boldsymbol{\sigma}, \Pi)$ for this completed history H_T.

The payoff for server j is $u_j = |C_j(H(\Pi))|$, the number of customers who join his queue. The (negative) payoff for customer c is $u_c = -\#\{c' \neq c \colon f(c') \in [a_c, f(c)]\}$. That is, c incurs a cost equal to the number of customers that are served ahead of c, but after c arrives, in the queue that c selects[3]. We refer to this cost as the *waiting time* of c. Write $u_i(\mathcal{A}_1, \mathcal{A}_2, \boldsymbol{\sigma}, \Pi)$ for utilities in the completed history $H(\mathcal{A}_1, \mathcal{A}_2, \boldsymbol{\sigma}, \Pi)$.

Equilibrium concepts A *safety-level equilibrium* (Ashlagi, Monderer, and Tennenholtz 2006) is a profile of strategies of the players, $\mathcal{A}_1, \mathcal{A}_2$, and $\boldsymbol{\sigma}$, such that, for all $c \in C$,

$$\sigma_c \in \operatorname*{argmax}_{\sigma'} \left\{ \min_{\Pi} \{u_c(\mathcal{A}_1, \mathcal{A}_2, (\sigma', \boldsymbol{\sigma}_{-c}), \Pi)\} \right\}$$

and for $\mathcal{A}_1$ (and similarly for $\mathcal{A}_2$),

$$\mathcal{A}_1 \in \operatorname*{argmax}_{\mathcal{A}'} \left\{ \min_{\Pi} \{u_1(\mathcal{A}', \mathcal{A}_2, \boldsymbol{\sigma}, \Pi)\} \right\}.$$

In other words, all players maximize their worst-case payoffs. Since players move sequentially, we focus on equilibria that are *subgame perfect*. Roughly speaking, a profile of

[1] Our results extend readily to infinite sequences of customers.

[2] We think of H as determining the state of each queue.

[3] Our use of this utility function, rather than the number of time units spent in queue, is motivated by the non-Bayesian setting.

strategies forms a subgame perfect equilibrium if, at every possible state of a partially completed game, the players' prescribed actions form an equilibrium for the remaining stages of the game. More formally, an equilibrium is *subgame perfect* for the customers if each customer c maximizes utility in the subgame that begins when the customer arrives, for every prior history H:

$$\sigma_c \in \operatorname*{argmax}_{\sigma'} \left\{ \min_{\Pi:\ H_{a_c}(\Pi)=H} \{u_c(\mathcal{A}_1, \mathcal{A}_2, (\sigma', \boldsymbol{\sigma}_{-c}), \Pi)\} \right\}.$$

An equilibrium is *subgame perfect* for the servers if, given a history H, the subsequent decisions made by each algorithm are utility-maximizing in the subgame beginning after H (Engelberg and Naor 2007). That is, for each $H \in \mathcal{H}$,

$$\mathcal{A}_1 \in \operatorname*{argmax}_{\mathcal{A}'} \left\{ \min_{\Pi:\ H \preceq H(\Pi, \mathcal{A}', \mathcal{A}_2, \boldsymbol{\sigma})} \{u_i(\mathcal{A}', \mathcal{A}_2, \boldsymbol{\sigma}, \Pi)\} \right\}$$

and similarly for $\mathcal{A}_2$. Note that a maximal $\mathcal{A}'$ must be consistent with H, and hence the optimization over $\mathcal{A}'$ is with respect to events following H. An equilibrium is subgame perfect if it is subgame perfect for the customers and servers.

Note that, given $\mathcal{A}_1$ and $\mathcal{A}_2$, there is a unique subgame perfect equilibrium for the customers[4], determined by backward induction. We will therefore often drop dependencies on $\boldsymbol{\sigma}$ from our notation, assuming customers apply subgame perfect strategies. We also write $u_j(\mathcal{A}_1, \mathcal{A}_2, H)$ for the utility of j if no further customers arrive after the events of H.

The competitive ratio of algorithm $\mathcal{A}_1$ against $\mathcal{A}_2$ is $CR(\mathcal{A}_1, \mathcal{A}_2) = \max_\Pi \max_{\mathcal{A}^*} \frac{u_1(\mathcal{A}^*, \mathcal{A}_2, \Pi)}{u_1(\mathcal{A}_1, \mathcal{A}_2, \Pi)}$. In a *competitive ratio equilibrium*, each server minimizes its competitive ratio given the strategies of the other players. That is, $\mathcal{A}_1 \in \operatorname{argmin}_{\mathcal{A}} CR(\mathcal{A}, \mathcal{A}_2)$ and similarly for $\mathcal{A}_2$.

3 Deterministic Safety-Level Equilibria

We begin by studying safety-level equilibria when schedulers are restricted to applying deterministic algorithms. The intuition we build will be of use when we consider the more technically demanding setting of randomized schedulers.

An alternative scheduler formulation

Formally, a scheduling algorithm determines how to select customers from queue, but an alternative (equivalent) formulation will be convenient. For each $H \in \mathcal{H}$, we will think of algorithm $\mathcal{A}_j$ as assigning to each $c \in Q_j(\mathcal{A}_j, H)$ a worst-case waiting time $s_c \in \mathbb{N} \cup \{\infty\}$, where the worst case is over $H' \succeq H$. We call s_c the *slot* held by c. Write $\mathbf{s}$ for a profile of slots. We say $\mathbf{s}$ is *valid* if, for each $k \geq 1$, at most k customers have slot $\leq k$ (this is a property of worst-case waiting times). Whenever a customer is served, an algorithm necessarily decreases each remaining customer's slot by 1, and an algorithm can never increase a customer's slot. This formulation is motivated by the following characterization of customer behavior, which follows immediately from the definition of subgame perfect safety-level equilibrium.

[4]For clarity of exposition, we will ignore issues of tie-breaking; it suffices that customers break ties in a publicly-known manner.

Algorithm 1 Promise Scheduler

State: Slot profile $\mathbf{s}$; guess $\mathbf{s}'$ at opponent's slot profile.

When a new customer c joins our queue:
1: **if** $w(\mathbf{s}) < w(\mathbf{s}')$ **then**
2: $\quad s_c \leftarrow \max\{s \in E(\mathbf{s}) : s < w(\mathbf{s}')\}$
3: **else** $s_c \leftarrow w(\mathbf{s})$

When the server is idle:
1: $c \leftarrow \operatorname{argmin}_c\{s_c\}$; Serve c
2: $s_c \leftarrow s_c - 1$ for all remaining customers

When a customer c joins the other queue:
1: $s'_c \leftarrow \max\{s \in E(\mathbf{s}') : s < w(\mathbf{s})\}$

Theorem 3.1. *Suppose $(\mathcal{A}_1, \mathcal{A}_2, \boldsymbol{\sigma})$ is a subgame perfect equilibrium. Then, for all $H \in \mathcal{H}$, $\sigma_c(H)$ is the queue that would assign the smaller slot to c after history H.*

Write $\mathbf{s}^j(\mathcal{A}_j, H)$ for the slot profile assigned by queue j, using $\mathcal{A}_j$, after history H. Slot k is *empty* in $\mathbf{s}$ if assigning a new customer to slot k would maintain validity. Write $E(\mathbf{s})$ for the set of empty slots, and $w(\mathbf{s}) = \min E(\mathbf{s})$. Let $n^k(\mathbf{s})$ be the number of empty slots in $\mathbf{s}$ less than or equal to k.

The Promise scheduler

We now describe the *Promise algorithm*, $\mathcal{P}$, listed as Algorithm 1. In addition to maintaining its slot profile $\mathbf{s}$, $\mathcal{P}$ maintains a profile for the other queue, $\mathbf{s}'$. An arriving customer is assigned the largest slot in $E(\mathbf{s})$ that is less than $w(\mathbf{s}')$ (or $w(\mathbf{s})$ if no such slot is available). The customer with minimal slot is always chosen to be served. Profile $\mathbf{s}'$ is maintained by assuming that the opponent behaves the same way: assigns incoming customers the largest slot in $E(\mathbf{s}')$ less than $w(\mathbf{s})$, and always serves the customer with minimal slot.

The main result in this section is the following.

Theorem 3.2. *$(\mathcal{P}, \mathcal{P})$ is the unique subgame-perfect safety-level equilibrium.*

Note Theorem 3.2 covers even non-symmetric and mixed strategy profiles. To prove Theorem 3.2, we begin with a lemma. We say that $\mathbf{s}$ *dominates* $\mathbf{s}'$ if $s_c \geq s'_c$ for all c.

Lemma 3.3. *If $H \preceq H(\mathcal{P}, \mathcal{P}, \Pi)$ and $H \preceq H(\mathcal{A}_1, \mathcal{P}, \Pi)$ for some $\mathcal{A}_1$, Π, and H, then $\mathbf{s}^1(\mathcal{P}, H)$ dominates $\mathbf{s}^1(\mathcal{A}_1, H)$.*

Proof. From the definition of $\mathcal{P}$, for each $c \in C_1(H)$, $s^1_c(\mathcal{P}, H)$ is maximal such that c would have chosen 1 given server 2 plays $\mathcal{P}$. Thus, for any $\mathcal{A}_1$ with $H \preceq H(\mathcal{A}_1, \mathcal{P}, \Pi)$, $s^1_c(\mathcal{A}_1, H) \leq s^1_c(\mathcal{P}, H)$ for all $c \in C_1(H)$. □

Lemma 3.4. *If $H(\mathcal{P}, \mathcal{P}, \Pi) \neq H(\mathcal{A}_1, \mathcal{P}, \Pi)$, then the first event at which they differ is a customer joining queue 1 in $H(\mathcal{P}, \mathcal{P}, \Pi)$ but joining queue 2 in $H(\mathcal{A}_1, \mathcal{P}, \Pi)$.*

Proof. Let H be the minimal common prefix. Since the next event after H is not common, it must be a customer c's choice of queue. Since $\mathbf{s}^1(\mathcal{P}, H)$ dominates $\mathbf{s}^1(\mathcal{A}_1, H)$ by Lemma 3.3, and $\mathcal{P}$ always offers a slot that would attract a

customer versus $\mathcal{P}$ if possible, it cannot be that c joins queue 1 in $H(\mathcal{A}_1, \mathcal{P}, \Pi)$ but not in $H(\mathcal{P}, \mathcal{P}, \Pi)$. □

We now show how Lemma 3.4 implies Theorem 3.2. We first show that $(\mathcal{P}, \mathcal{P})$ is a subgame perfect equilibrium. For all $H \in \mathcal{H}$ and all $\mathcal{A}_1, \mathcal{A}_2$, the minimum payoff $\min_{\Pi: H \preceq H(\Pi)}\{u_1(\mathcal{A}_1, \mathcal{A}_2, \Pi)\}$ occurs when no further customers arrive following H. Subgame perfection therefore corresponds to always attracting an incoming customer whenever there exists an algorithm consistent with H that does so. Lemma 3.4 therefore immediately implies that $\mathcal{P}$ is subgame perfect when played against $\mathcal{P}$, as required.

We next show $(\mathcal{P}, \mathcal{P})$ is the unique subgame perfect equilibrium. Suppose $(\mathcal{A}_1, \mathcal{A}_2)$ is subgame perfect with $\mathcal{A}_1 \neq \mathcal{P}$ or $\mathcal{A}_2 \neq \mathcal{P}$. Then $\exists \Pi$ with $H(\mathcal{A}_1, \mathcal{A}_2, \Pi) \neq H(\mathcal{P}, \mathcal{P}, \Pi)$. Let H be the minimal common prefix, and choose Π so that H is shortest. Then, without loss of generality, the next event in $H(\mathcal{A}_1, \mathcal{A}_2, \Pi)$ and $H(\mathcal{P}, \mathcal{P}, \Pi)$ is a customer c joining queue 2 in the former, but queue 1 in the latter. Let Π' be Π excluding all customers that have not arrived in H, but including c. Then $u_1(\mathcal{A}_1, \mathcal{A}_2, \Pi') < u_1(\mathcal{P}, \mathcal{P}, \Pi')$.

Consider $H(\mathcal{P}, \mathcal{A}_2, \Pi')$. If $H \not\preceq H(\mathcal{P}, \mathcal{A}_2, \Pi')$, then by Lemma 3.4 the first point of difference between H and $H(\mathcal{P}, \mathcal{A}_2, \Pi')$ must be a customer joining 2 in H but not in $H(\mathcal{P}, \mathcal{A}_2, \Pi')$ (since H is a prefix of $H(\mathcal{P}, \mathcal{P}, \Pi')$). This contradicts the supposed subgame perfection of $\mathcal{A}_2$. So we must have $H \preceq H(\mathcal{P}, \mathcal{A}_2, \Pi')$. Then, since c joins queue 1 in $H(\mathcal{P}, \mathcal{P}, \Pi')$, by Lemma 3.4 it must also join queue 1 in $H(\mathcal{P}, \mathcal{A}_2, \Pi')$, and hence $u_1(\mathcal{P}, \mathcal{A}_2, \Pi') = u_1(\mathcal{P}, \mathcal{P}, \Pi) > u_1(\mathcal{A}_1, \mathcal{A}_2, \Pi')$. Since both $\mathcal{P}$ and $\mathcal{A}_1$ are consistent with history H, we conclude that $\mathcal{A}_1$ does not maximize utility in the subgame beginning after history H, contradicting subgame perfection. This completes the proof of Theorem 3.2.

4 Randomized Safety-Level Equilibria

We now consider games with randomized scheduling algorithms. We again find a unique subgame perfect safety-level equilibrium. Our equilibrium scheduler generalizes the Promise scheduler by allowing lotteries over slots, and tracking the *expected* slot assigned to each customer in queue.

A characterization of randomized schedulers

It will again be convenient to view schedulers as assigning wait times to customers. Since a randomized algorithm $\mathcal{A}$ is a convex combination of deterministic algorithms, its state can be described by a distribution Γ over slot assignments to customers. A distribution is *valid* if each slot assignment in its support is valid. Given valid distribution Γ and customer c, the *expected slot* of c is $e_c = \mathbf{E}_{\mathbf{s}\sim\Gamma}[s_c]$. Note that $e_c \in \mathbb{R}_{\geq 1} \cup \{\infty\}$. Write $\mathbf{e}$ for a profile of expected slots, and let $C(\mathbf{e})$ denote the customers represented in $\mathbf{e}$. The definition of subgame perfection implies that each customer chooses the queue that offers the smaller expected slot.

Claim 4.1. *Fix scheduling algorithms $\mathcal{A}_1$, $\mathcal{A}_2$. The unique subgame perfect safety-level equilibrium for the customers is for each customer c to select the queue that minimizes the expected slot that would be assigned to c.*

By Claim 4.1, the outcome of the game is uniquely determined by the input profile Π and server strategies $\mathcal{A}_1, \mathcal{A}_2$. It also motivates us to study which profiles of expected slots are realizable by valid distributions. A profile $\mathbf{e}$ is *valid* if it is realizable by a valid distribution Γ. The following result characterizes all valid expected slot assignments. This lemma turns out to be a restatement of a classical result from scheduling theory (Horvath, Lam, and Sethi 1977), expressed in the context of ad auctions by (Feldman et al. 2008). Let $[\mathbf{e}]_i$ denote the ith-smallest entry of $\mathbf{e}$.

Lemma 4.2. *Expected slot profile $\mathbf{e}$ is valid if and only if, for all $k \leq n$, $\sum_{i=1}^{k}[\mathbf{e}]_i \geq \sum_{i=1}^{k} i$.*

Given Lemma 4.2, we think of a scheduler as assigning an expected slot (i.e. expected worst-case wait time) to each customer, subject to the validity condition. Moreover, if $\mathbf{e}$ is valid, then so is $\mathbf{e}'$ whenever $[\mathbf{e}]'_i \geq [\mathbf{e}]_i$ for every i. Given valid $\mathbf{e}$, there is a minimal z such that $\mathbf{e}$ would remain valid if a new customer c was admitted and assigned[5] $e_c = z$. Let $w(\mathbf{e})$ be this minimal value z. We are interested in a notion of one slot profile being "better" than another, from the perspective of maximizing worst-case server utility.

Definition 1. *Expected slot profile* $\mathbf{e}$ dominates $\mathbf{e}'$ *if* $\sum_{i=1}^{k}[\mathbf{e}]_i \geq \sum_{i=1}^{k}[\mathbf{e}]'_i$ *for all* $k \geq 1$.

Claim 4.3. *If $\mathbf{e}$ dominates $\mathbf{e}'$ then $w(\mathbf{e}) \leq w(\mathbf{e}')$.*

Proof. Note that $w(\mathbf{e})$ is precisely the minimal value such that $\sum_{i=1}^{k-1}[\mathbf{e}]_i + w(\mathbf{e}) \geq \sum_{i=1}^{k} i$ for all k, and similarly for $w(\mathbf{e}')$. If $z \geq w(\mathbf{e}')$ then $\sum_{i=1}^{k-1}[\mathbf{e}]'_i + z \geq \sum_{i=1}^{k} i$ for all k. But then $\sum_{i=1}^{k-1}[\mathbf{e}]_i + z \geq \sum_{i=1}^{k} i$, and hence $z \geq w(\mathbf{e})$. □

The probabilistic promise scheduler

We now describe subgame perfect safety-level equilibrium over randomized algorithms. The *(probabilistic) promise algorithm*, $\mathcal{P}^P$, is listed as Algorithm 2. Informally, $\mathcal{P}^P$ is similar to $\mathcal{P}$: it maintains a slot profile for itself, $\mathbf{e}$, as well as for the opponent, $\mathbf{e}'$. Whenever a new customer joins the queue, it is assigned the largest expected slot greater than $w(\mathbf{e})$ but smaller than $w(\mathbf{e}')$, if any; otherwise it is assigned $w(\mathbf{e})$. To deal with issues of tie-breaking, we allow the scheduler to offer an expected wait time of $w(\mathbf{e}')^-$, defined to be infinitesimally smaller than $w(\mathbf{e}')$. One can view such infinitesimals as implementing a tie-breaking rule for the customers, arising in the limit of ϵ-approximate equilibria as ϵ tends to 0; see (Ashlagi et al. 2010) for a discussion.

Whenever a customer is to be served, the algorithm proceeds as follows. First, it chooses the customers that will be selected with positive probability: those with the k smallest expected slots (line 2), for some k (described below). Each of these customers is assigned a probability p_i with which they will be served (line 3). A customer is then chosen for service according to this distribution (line 4). The remaining $k-1$ of these k customers have their expected slots set to a certain quantity $X = \frac{Z-1}{k-1}$, where $Z = \sum_{i=1}^{k}[\mathbf{e}]_i$ (line 5). The values p_i are chosen so that each customer's

[5] There is a minimal z, rather than an infinimum, because the inequalities of Lemma 4.2 are not strict.

Algorithm 2 Probabilistic Promise Scheduler

State: Exp. slot profile $\mathbf{e}$; guess $\mathbf{e}'$ at opponent's profile.

When a new customer c joins our queue:

1: **if** $w(\mathbf{e}) < w(\mathbf{e}')$ **then** $e_c \leftarrow w(\mathbf{e}')^-$
2: **else** $e_c \leftarrow w(\mathbf{e})$

When the server is idle:

1: **if** $|C(\mathbf{e})| > 1$ and $[\mathbf{e}]_1 > 1$ **then**
2: Let $k \geq 1$ be min s.t. $(\sum_{i=1}^{k}[\mathbf{e}]_i) - 1 \leq (k-1)[\mathbf{e}]_{k+1}$
3: Let $Z \leftarrow \sum_{i=1}^{k}[\mathbf{e}]_i$, $p_i \leftarrow \frac{Z-1-(k-1)[\mathbf{e}]_i}{Z-k}$ for each i
4: Choose $i \in [k]$ from distribution defined by p_i's
5: $[\mathbf{e}]_i \leftarrow 1$; $[\mathbf{e}]_j \leftarrow (Z-1)/(k-1)\ \forall\, j \leq k, j \neq i$
6: $c \leftarrow \text{argmin}\{e_c\}$; Serve c; Remove c from $C(\mathbf{e})$
7: $e_c \leftarrow e_c - 1\ \forall\, c \in C(\mathbf{e})$

When a customer c joins the other queue:

1: $e'_c \leftarrow w(\mathbf{e})^-$

expected wait time is invariant under this procedure, i.e. $[\mathbf{e}]_i = p_i \cdot 1 + (1-p_i)\cdot X$. The value k is chosen to be the smallest so that, when the quantities are set as described above, we will have $X \leq [\mathbf{e}]_{k+1}$, so that the relative order of the customers' slots is unchanged.

To maintain state $\mathbf{e}'$ we assume that, when a customer joins the opponent's queue, it is assigned slot $w(\mathbf{e})^-$, and that customers are served as described above.

Lemma 4.4. *$\mathcal{P}^P$ is a feasible scheduling algorithm.*

Proof. We first show that the values p_i (from line 3) form a distribution. By validity, $[\mathbf{e}]_i \geq 1$ for all i, so $Z \geq k$. Thus, by the choice of k, we have

$$\begin{aligned} p_i &\geq Z - 1 - (k-1)[\mathbf{e}]_k \\ &> ((k-2)[\mathbf{e}]_k + [\mathbf{e}]_k + 1) - 1 - (k-1)[\mathbf{e}]_k \geq 0. \end{aligned}$$

Noting also that $\sum_i p_i = 1$ establishes well-definedness.

We now show that validity is maintained after performing the modifications on lines 5. Let $\mathbf{e}$ and $\overline{\mathbf{e}}$ denote the expected slot assignments before and after line 5. For all $j < k$,

$$\sum_{i \leq j}[\overline{\mathbf{e}}]_i = 1 + (j-1)\frac{Z-1}{k-1} \geq \sum_{i \leq j}[\mathbf{e}]_i \geq \sum_{i \leq j} i.$$

For all $j \geq k$, we have $\sum_{i \leq j}[\overline{\mathbf{e}}]_i = \sum_{i \leq j}[\mathbf{e}]_i \geq \sum_{i \leq j} i$, since the sum of the lowest k expected slots is unchanged.

We next show validity is maintained after line 7. Let $\mathbf{e}$ and $\tilde{\mathbf{e}}$ denote the expected slots before and after line 7. For all k,

$$\sum_{i=1}^{k}[\tilde{\mathbf{e}}]_i = \sum_{i=2}^{k+1}([\mathbf{e}]_i - 1) \geq \left(\sum_{i=1}^{k+1} i\right) - k - [\mathbf{e}]_1 = \sum_{i=1}^{k} i.$$

Finally, to show that e_c is indeed the worst-case expected wait time for customer c, note first that if $e_c = 1$ then c is served next. Otherwise, if c is is a possible candidate to be served on an interation of the algorithm, then his expected wait time over the algorithm's randomness is

$$p_c \cdot 1 + (1-p_c)\cdot\frac{Z-1}{k-1} = e_c.$$

Finally, each customer who is not served has its expected wait time reduced by 1, as is required. □

We next show that the method used to select customers from queue is optimal: it generates an expected slot assignment that dominates the assignment of any other method.

Lemma 4.5. *Fix valid expected slot profile $\mathbf{e}$, and let $\mathbf{e}^{\mathcal{P}^P}$ denote the profile that results after a customer is selected from queue according to $\mathcal{P}^P$. Consider any other expected slot assignment $\tilde{\mathbf{e}}$ that results from a customer selection method respecting $\mathbf{e}$. Then $\mathbf{e}^{\mathcal{P}^P}$ dominates $\tilde{\mathbf{e}}$.*

Proof. Let $m = |C(\mathbf{e})|$ and consider any customer selection method that respects $\mathbf{e}$. This assigns to each $c \in C(Q)$ a probability p_c of being served, and an expected wait time $\tilde{e}_c$ if c is not served, where $p_c \cdot 1 + (1-p_c)\tilde{e}_c \leq e_c$.

Note that $\tilde{\mathbf{e}}$ is this profile of expected slots $\tilde{e}_c$, excluding the customer who was served. If the customer with k-lowest expected slot in $\mathbf{e}$ is served, then $[\tilde{\mathbf{e}}]_i \geq [\mathbf{e}]_i$ for each $i < k$ and $[\tilde{\mathbf{e}}]_i \geq [\mathbf{e}]_{i+1}$ for $i \geq k$; this follows since $[\mathbf{e}]_i \geq 1$ for all i. For any such profile $\tilde{\mathbf{e}}$, we claim that

$$1 + \sum_{i=1}^{m-1}[\tilde{\mathbf{e}}]_i \leq \sum_{i=1}^{m}[\mathbf{e}]_i. \tag{1}$$

To see this, note that

$$\sum_{i=1}^{m}[\mathbf{e}]_i = \sum_{c \in C(\mathbf{e})}(p_c + (1-p_c)\tilde{e}_c) = 1 + \sum_{c \in C(\mathbf{e})}(1-p_c)\tilde{e}_c$$

where the coefficients $(1-p_c)$ each lie in $[0,1]$ and sum to $(m-1)$. The RHS is therefore minimized when $p_c = 1$ for some c maximizing $\tilde{e}_c$, from which we conclude (1).

Recall $\mathcal{P}^P$ selects k, sets $[\mathbf{e}^{\mathcal{P}^P}]_j = \frac{1}{k-1}((\sum_{i=1}^{k}[\mathbf{e}]_i) - 1)$ for $j < k$, and $[\mathbf{e}^{\mathcal{P}^P}]_j = [\mathbf{e}]_{j+1}$ for $j \geq k$. Value k is chosen to be minimal such that $[\mathbf{e}^{\mathcal{P}^P}]_{k-1} \leq [\mathbf{e}]_{k+1}$.

Suppose for contradiction that there exists some ℓ such that $\sum_{i=1}^{\ell}[\tilde{\mathbf{e}}]_i > \sum_{i=1}^{\ell}[\mathbf{e}^{\mathcal{P}^P}]_i$. We consider two cases, depending on whether $\ell < k$ or $\ell \geq k$. If $\ell < k$, then $\sum_{i=1}^{\ell}[\mathbf{e}^{\mathcal{P}^P}]_i = \frac{\ell}{k-1}((\sum_{i=1}^{k}[\mathbf{e}]_i) - 1)$. But now $\sum_{i=1}^{\ell}[\tilde{\mathbf{e}}]_i > \frac{\ell}{k-1}((\sum_{i=1}^{k}[\mathbf{e}]_i) - 1)$ implies $[\tilde{\mathbf{e}}]_j > \frac{1}{k-1}((\sum_{i=1}^{k}[\mathbf{e}]_i) - 1)$ for all $\ell < j < k$, since the $[\tilde{\mathbf{e}}]_j$ are non-decreasing. Furthermore, $[\tilde{\mathbf{e}}]_j \geq [\mathbf{e}]_{j+1}$ for all $j \geq k$. Putting this together,

$$\sum_{i=1}^{m-1}[\tilde{\mathbf{e}}]_i > \left(\left(\sum_{i=1}^{k}[\mathbf{e}]_i\right) - 1\right) + \sum_{i=k+1}^{m}[\mathbf{e}]_i = \sum_{i=1}^{m}[\mathbf{e}]_i - 1$$

contradicting (1). Next suppose that $\ell \geq k$. Then

$$\sum_{i=1}^{\ell}[\mathbf{e}^{\mathcal{P}^P}]_i = \left(\sum_{i=1}^{k}[\mathbf{e}]_i\right) - 1 + \sum_{i=k+1}^{\ell+1}[\mathbf{e}]_i = \left(\sum_{i=1}^{\ell+1}[\mathbf{e}]_i\right) - 1.$$

We then have that

$$\sum_{i=1}^{m-1}[\tilde{\mathbf{e}}]_i > \sum_{i=1}^{\ell}[\mathbf{e}^{\mathcal{P}^P}]_i + \sum_{i=\ell+2}^{m}[\mathbf{e}]_i = \sum_{i=1}^{m}[\mathbf{e}]_i - 1$$

contradicting (1). □

We can now show that $\mathcal{P}^P$ is our desired equilibrium.

Theorem 4.6. *$(\mathcal{P}^P, \mathcal{P}^P)$ is the unique subgame perfect safety-level equilibrium.*

The proof of Theorem 4.6 is similar to Theorem 3.2. The space of potential strategies is larger, but Claim 4.3 and Lemma 4.5 imply that, on any history H consistent with $(\mathcal{P}^P, \mathcal{P}^P)$ and any algorithm $\mathcal{A}$ consistent with H when played by queue j, the state of $\mathcal{P}^P$ dominates that of $\mathcal{A}$. Given this, we can apply the arguments from Theorem 3.2 to show that $(\mathcal{P}^P, \mathcal{P}^P)$ is the unique subgame perfect equilibrium. The details appear in the full version of the paper.

5 Deterministic Competitive Ratio Equilibria

We now consider competitive ratio equilibria over the space of deterministic algorithms. We show that even though the FCFS scheduler, $\mathcal{F}$, is not a symmetric subgame perfect safety-level equilibrium (Theorem 3.2), $(\mathcal{F}, \mathcal{F})$ does form a competitive ratio equilibrium. Specifically, we prove that $\mathcal{F}$ has competitive ratio 2 against $\mathcal{F}$, and that this is the best possible competitive ratio that can be achieved against $\mathcal{F}$.

Theorem 5.1. *$(\mathcal{F}, \mathcal{F})$ forms a competitive ratio equilibrium, with competitive ratio 2.*

Proof sketch. We first show $CR(\mathcal{F}, \mathcal{F}) \leq 2$. Choose algorithm $\mathcal{A}$ and input Π. Write $H^{\mathcal{A}} = H(\mathcal{A}, \mathcal{F}, \Pi)$ and $H^{\mathcal{F}} = H(\mathcal{F}, \mathcal{F}, \Pi)$. Write $H_t^{\mathcal{A}}$ and $H_t^{\mathcal{F}}$ for the prefixes of $H^{\mathcal{A}}$ and $H^{\mathcal{F}}$ up to time t. Recall from Section 3 the definition of $\mathbf{s}^j(\mathcal{A}_j, H)$. Define the *potential* of a pair of prefixes, $f(H_t^{\mathcal{A}}, H_t^{\mathcal{F}})$, to be the number of slots empty in $\mathbf{s}^1(\mathcal{A}, H_t^{\mathcal{A}})$ but not $\mathbf{s}^1(\mathcal{F}, H_t^{\mathcal{F}})$, plus the number of slots empty in $\mathbf{s}^2(\mathcal{F}, H_t^{\mathcal{F}})$ but not $\mathbf{s}^2(\mathcal{A}, H_t^{\mathcal{A}})$. A straightforward case analysis (given in the full version) shows that $|C_1(H^{\mathcal{A}})| + f(H_t^{\mathcal{A}}, H_t^{\mathcal{F}})$ increases only if some $c \in C$ joins queue 1 in $H^{\mathcal{F}}$, and by at most 2. This implies $|C_1(H_t^{\mathcal{A}})| + f(H_t^{\mathcal{A}}, H_t^{\mathcal{F}}) \leq 2|C_1(H_t^{\mathcal{F}})|$ at each t. Since $f(H^{\mathcal{A}}, H^{\mathcal{F}}) = 0$, $|C_1(H^{\mathcal{A}})| \leq 2|C_1(H^{\mathcal{F}})|$ as required

We next show that the competitive ratio of 2 is optimal. Choose $\mathcal{A}$ and arbitrarily large $R > 0$. Assume for contradiction that $CR(\mathcal{A}, \mathcal{F}) \leq 2 - \frac{1}{R}$. We will describe an input Π by describing $H(\mathcal{A}, \mathcal{F}, \Pi)$. Begin with $2R$ customer arrivals; let H_0 be the history after these arrivals. Let $T = |C_2(\mathcal{A}, \mathcal{F}, H_0)|$. Note $T \geq R$ from the definition of $\mathcal{F}$. We therefore have $|C_1(\mathcal{A}, \mathcal{F}, H_0)| \leq T$, and each $c \in C_1(\mathcal{A}, \mathcal{F}, H_0)$ is assigned a slot at most T. We will show by induction that, for all $i \geq 1$, there is a valid history $H_i \succeq H_0$ such that $|C_1(\mathcal{A}, H_i)| \geq |C_1(\mathcal{A}, H_0)| + i$, with each customer in $C_1(\mathcal{A}, H_i)$ assigned a slot at most T, and $C_2(\mathcal{F}, H_i)$ has T customers in slots 1 through T. For $i > T$ the slot assignment of queue 1 is invalid, a contradiction.

For the induction, suppose H_i ends at time t, $|C_1(\mathcal{A}, H_i)| \geq 2R - T + i$, and all $c \in C_1(\mathcal{A}, H_i)$ are assigned to slots at most T. Let $S = |C_1(\mathcal{A}, H_i)|$, so $S \leq T$ by validity. Define $H_i' \succeq H_i$ as follows: alternate between serving a customer from queue 2 and having a new customer arrive, $2T$ times. Suppose for contradiction that none of these new customers join server 1. Let Π be the set of customers that arrive in H_i'. We then have that $u_1(\mathcal{A}, \mathcal{F}, \Pi) = S$. Let $\mathcal{A}'$ be an algorithm that offers each customer in H_i a slot of ∞, then offers to each subsequent customer the maximum available slot less than the slot offered by server 2 (assuming that server 2 uses $\mathcal{F}$). Then, at time t, queue 2 has $S + T$ customers in slots 1 through $S + T$. Each of the next $S + T - 1$ customers arriving in H_i' join server 1, as the kth customer would be offered slot $S+T-k$. We therefore have $u_1(\mathcal{A}', \mathcal{F}, \Pi) \geq S+T-1$. Since $S \leq T = 2R$, this implies a competitive ratio of at least $2 - \frac{1}{R}$, a contradiction.

We conclude that a customer joins queue 1 in H_i'. Let $H_{i+1} \preceq H_i'$ be the history ending after the first customer joins server 1. Then $|C_1(\mathcal{A}, H_{i+1})| = |C_1(\mathcal{A}, H_i)| + 1 \geq 2R - T + i + 1$, and moreover $|C_2(\mathcal{F}, H_{i+1})| = |C_2(\mathcal{F}, H_i)| = T$. This completes the induction. □

Competitive Ratio of the Promise Scheduler

We next consider the competitive ratio of the promise scheduler, $\mathcal{P}$. We find that $CR(\mathcal{P}, \mathcal{P}) > 5/2$. Thus, while profile $(\mathcal{P}, \mathcal{P})$ is unique among subgame perfect safety-level equilibria, it is strictly worse than profile $(\mathcal{F}, \mathcal{F})$ with respect to the decision criterion of competitive ratio.

Proposition 5.2. $CR(\mathcal{P}, \mathcal{P}) \geq \frac{5}{2}$.

The proof of Proposition 5.2 is similar to the second half of Theorem 5.1 (which showed $CR(\mathcal{F}, \mathcal{F}) \geq 2$). The additional idea that results in an improved bound is that a scheduler facing an opponent using $\mathcal{P}$ can sometimes benefit by strategically leaving certain slots open (i.e., by sacrificing customers). The reason this is useful against an opponent playing $\mathcal{P}$ is that such open slots can affect the slots offered by his opponent to future customers. We construct an input instance in which an algorithm with future knowledge can apply such a strategy to cause his opponent to fill his lowest slots, leading to a significant increase future customers. The details appear in the full version of the paper.

6 Conclusion

We analyzed a competition between online scheduling algorithms with strategic customers and observable queues. We studied the non-Bayesian equilibrium concept of safety-level equilibrium, and found that the unique equilibrium occurs when both schedulers apply the novel *Promise* algorithm. If we instead take the view from online algorithm analysis that schedulers wish to maximize competitive ratio, then it is an equilibrium for both schedulers to apply FCFS.

This work provides more questions than answers; we list just a few future directions. A natural step would be to extend our results to 3 or more servers. One might also explore Bayesian settings, e.g. Possion arrival rates and exponential service times, or non-observable queues. Perhaps servers could choose the information revealed to the customers (and/or opponent)[6]. One might introduce heterogeneity between servers, e.g. in quality or speed of service. We leave open the question of whether FCFS is the *optimal* competitive ratio equilibrium (in terms of the competitive ratio achieved). Finally, we suspect the general framework of equilibria for online algorithms with strategic users can be applied to other algorithmic settings.

[6] See (Hassin 1986) for single-server information decisions.

References

Alperstein, H. 1988. Optimal pricing for the service facility offering a set of priority prices. *Management Science* 34:666–671.

Ashlagi, I.; Braverman, M.; Hassidim, A.; Lavi, R.; and M.Tennenholtz. 2010. Position auctions with budgets: Existence and uniqueness. *B.E. Journal of Theoretical Economics - Advances* 10(1)).

Ashlagi, I.; Monderer, D.; and Tennenholtz, M. 2006. Resource selection games with unknown number of players. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, 819–825.

Ashlagi, I.; Monderer, D.; and Tennenholtz, M. 2009. Two-terminal routing games with unknown active players. *Artificial Intelligence Journal* 173(15):1441–1455.

Ashlagi, I.; Tennenholtz, M.; and Zohar, A. 2010. Competing schedulers. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)*.

Burguet, R., and Sakovics, J. 1999. Imperfect Competition in Auction Designs. *International Economic Review* 40(1):231–247.

Engelberg, R., and Naor, J. 2007. Equilibria in Online Games. In *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 149–158.

Feldman, J.; Muthukrishnan, S.; Nikolova, E.; and Pál, M. 2008. A truthful mechanism for offline ad slot scheduling. In *Proceedings of the 1st International Symposium on Algorithmic Game Theory*, SAGT '08, 182–193. Berlin, Heidelberg: Springer-Verlag.

Hassin, R. 1985. On the optimality of first-come last-served queues. *Econometrica* 53:201–202.

Hassin, R. 1986. Consumer information in markets with random products quality: The case of queues and balking. *Econometrica* 54:1185–1195.

Hassin, R. 2009. Equilibrium customers choice between FCFS and Random servers. *Queueing Systems* 62:243–254.

Horvath, E. C.; Lam, S.; and Sethi, R. 1977. A level algorithm for preemptive scheduling. *J. ACM* 24(1):32–43.

Immorlica, N.; Kalai, A.; Moitra, A.; Postlewaite, A.; and Tennenholtz, M. 2011. Dueling Algorithms. In *Proceedings of the 43th Symposium on Theory of Computing*.

Immorlica, N.; Kleinberg, R.; and Mahdian, M. 2006. Secretary problems with competing employers. In *Internet and Network Economics, LNCS 4286*, 389–400.

McAfee, P. 1993. Mechanism Design by Competing Sellers. *Econometrica* 61:1281–1312.

Naor, P. 1969. The regulation of queue size by levying tolls. *Econometrica* 37:15–24.

Peters, M., and Severinov, S. 1997. Competition Among Sellers Who Offer Auctions Instead of Prices. *Joutnal of Ecoonimc Theory* 75:141–179.