Bounding the Power of Preemption in Randomized Scheduling

Ran Canetti[1]
Sandy Irani[2]

Technical Report 95-07
Department of Information and Computer Science
University of California, Irvine

March 1995

[1]Deptartment of Applied Mathematics and Computer Science
Weizmann Institute, Israel.

[2]Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

Keywords: Online algorithms, multiprocessor scheduling, virtual circuit routing

The work of the first author was supported by the American-Israeli Binational Science Foundation grant 92-00226 at the Weizmann Institute, Israel and Columbia University. The work of the second author was supported by National Science Foundation Grants CCR-9309456 and GER-94-50142 at Univerity of California, Irvine and Columbia University.

Bounding the Power of Preemption in Randomized Scheduling

Ran Canetti* Sandy Irani†

March 22, 1995

Abstract

We study on-line scheduling in overloaded systems. Requests for jobs arrive one by one as time proceeds; the serving agents have limited capacity and not all requests can be served. Still, we want to serve the 'best' set of requests according to some criterion. In this situation, the ability to **preempt** (i.e., abort) jobs in service in order to make room for better jobs that would otherwise be rejected has proven to be of great help in some scenarios.

We show that, surprisingly, in many other scenarios this is not the case. In a simple, generic model, we prove a polylogarithmic lower bound on the competitiveness of randomized and preemptive on-line scheduling algorithms. Our bound applies to several recently studied problems. In fact, in certain scenarios our bound is quite close to the competitiveness achieved by known *deterministic, non-preemptive* algorithms.

1 Introduction

Scheduling problems pervade many aspects in system design and management. Consider, for instance, the following problems:

(a). A system with several processors is assigned jobs of varying duration and load, where each job is to be performed on a single processor. The jobs arrive one by one and each job must be assigned to a processor before the next job is known. The goal is to assign the jobs to processors "in the best possible way".

(b). A set of gateways connects a network of computers to a set of peripheral devices. At any point in time, a node in the network may request a connection to a particular type of device for

*Dept. of Applied Mathematics and Computer Science, Weizmann Institute, Israel. Supported by the American-Israeli Binational Science Foundation grant 92-00226. Email: canetti@wisdom.weizmann.ac.il.

†Supported in part by NSF Grants CCR-9309456 and GER-94-50142. Address: Department of Information and Computer Science, University of California, Irvine, CA 92717. Email: irani@ics.uci.edu .

some period of time. The bandwidth required for the different connections vary. The goal is, again, to assign connections to gateways "in the best possible way".
(c). A communication network with guaranteed bandwidth policy services many different types of traffic. Requests for connections between nodes in the network arrive and depart through time. The durations, priorities, and bandwidth requirements of connections vary greatly (e.g. e-mail, video, etc.). Connections should be allocated bandwidth and routed "in the best possible way". We note that this problem is currently of great practical value. High speed communication networks, which are the communication backbone of the future, have guaranteed bandwidth policy and are confronted with such bandwidth allocation problems (e.g., ATM [22]).

A prominent characteristic of all these scenarios is that the scheduling algorithm learns about the incoming requests as they enter the system one by one, and must decide how to handle each task without knowledge of future tasks.

Two natural optimization problems arise from these and other related scenarios. First, we may assume that all requests are served and aim at minimizing the maximum load on any serving agent (e.g., processor, gateway, link) at any point in time. We call this problem **load balancing**. Alternatively, the capacity of the system may be limited, thus not all requested can be served. Now the goal is to schedule the subset of requests with maximum value according to some criterion. We call this problem **admission control**. Both optimization problems have been studied in several models. The load balancing problem emerging from example (a) above is an old problem introduced in [14] and extensively studied since [1, 9, 15]. The admission control problem emerging from this example is studied in [10, 11, 21]. The load balancing problem emerging from example (b) is studied in [5, 7] and of example (c) in [1]. The admission control problem emerging from example (c) is studied in [2]. In this work we address admission control problems. (We refer the reader to [10] for an exposition of the importance of admission control.)

A natural question in such scenarios has to do with the power of **preemption**. The ability to alter previous decisions in certain ways has proven to be very powerful (say, when algorithms are allowed to reassign a job to a different server at some cost [19, 18]). We address the following notion of preemption, natural in limited capacity scenarios. We allow a job to be aborted in the middle of execution, in order to make room for a more valuable job that would otherwise be rejected. However, no credit is accrued for uncompleted jobs. Preempting jobs in the middle of execution may be problematic in some scenarios (say, when service is guaranteed upon admission). In other scenarios, preemption seems to be acceptable and even very helpful (say, in systems that support real-time jobs, e.g. [20]).

It has been demonstrated that an appropriate use of preemption helps considerably to enhance the **throughput** in certain settings [8]. We study the following question. To what extent can the ability to preempt jobs enhance the performance in more general cases (e.g., when the criteria for performance are different, or when the setting is different)?

We provide some surprising answers to this question by demonstrating that preemption does not help much (if at all) in a large variety of on-line admission control problems. First we consider a generic, simple model for on-line admission control. In this model we show a lower bound on the competitiveness of any *randomized, preemptive* on-line scheduling algorithm. This bound applies, via simple reductions, to a large variety of models, and in particular to the admission control problems emerging from the three examples above. We elaborate on the different scenarios where our result applies in section 1.1.

We describe our generic model. A server is given a sequence of job requests, arriving one by

one as time proceeds. The server can serve only one job at a time. Each job is characterized by its arrival time, its **duration** (known upon arrival), and its **value**. A job has to be either rejected or served immediately for the specified duration. The server can **preempt** (i.e., abort) jobs in service. The value gained from a sequence is the sum of the values of completed jobs. No value is gained for preempted jobs. We consider randomized scheduling algorithms (or **schedulers**). A scheduler is f-competitive if for any sequence S of jobs the value gained by the best (off-line) schedule on S divided by the value gained by the online scheduler on S is at most f. Note that f may be a function of the request sequence, rather than a constant. In this model, we prove the following bound:

Theorem 1 *Any randomized, preemptive scheduler has a competitive ratio of at least*

$$\Omega\left(\sqrt{\frac{\log \mu}{\log \log \mu}}\right)$$

where μ is any of the following two measures:

(i) *$\mu = \mu_v$, the ratio between the largest and smallest value of a job in the request sequence.*

(ii) *$\mu = \mu_d$, the ratio between the largest and smallest duration of a job in the request sequence.*

We actually prove a stronger fact than suggested by the theorem: for any randomized preemptive scheduler $\mathcal{A}$ and for any μ, we construct a sequence S which simultaneously satisfies $\mu_d(S) \leq \mu$ and $\mu_v(S) \leq \mu$, such that the competitiveness of $\mathcal{A}$ on S is $\Omega(\sqrt{\log \mu / \log \log \mu})$. An identical bound applies also to scenarios where any fixed number of jobs can be served at a time. The bound does not depend in any way on the number of jobs which can be served at a time. In Section 5 we describe a simple, randomized, preemptive $O(\log(\min\{\mu_d, \mu_v\}))$-competitive scheduler in our generic model, thus demonstrating that our bound is at most a roughly quadratic factor from optimality.

Many previously studied scheduling problems can be reduced to this generic model. In particular, our bound applies to the setting in which Awerbuch Azar and Plotkin show a scheduler with competitiveness logarithmic in μ_d and μ_v [2]. Their scheduler is both *deterministic* and *non-preemptive*. Thus, in their setting, the combined power of randomization and preemption results in at most a roughly quadratic improvement. (We note, however, that the scheduler in [2] does not apply to the setting of Theorem 1.)

Our proof of the bound is non-trivial and occupies most of this paper. It involves techniques which we believe are of independent interest. For each randomized scheduler, we construct a request sequence on which the scheduler performs poorly relative to the best strategy. The sequence does not depend on the random choices of the scheduler. The sequence is constructed one request at a time via an interaction with the scheduler, $\mathcal{A}$. Each next request is generated based on the probability distribution of the job $\mathcal{A}$ currently has in service. That is, at each step the adversary keeps a set of threshold values. The adversary generates the next request depending on whether the probability that $\mathcal{A}$ has some specific job in service is above or below some threshold. The adversary strategy is a randomized adaptation of a simple deterministic lower bound. Our technique may prove useful in transforming similar deterministic lower bounds into randomized ones.

1.1 Applications of the bound

We describe how our lower bound applies to recently studied admission control problems. In Section 4 we formally state and prove these applications.

Consider the following admission control problem, emerging from example (a) above. A system consists of a set of processors, each with limited capacity. Jobs of varying load, duration and value arrive through time, each with a deadline. At all times the sum of the loads of the jobs in service on a given processor must not exceed its capacity. A scheduler must decide which jobs to execute (and on which processor) in order to complete the set of jobs with the largest total value before their deadlines. Via a simple reduction, our lower bound of $\Omega(\sqrt{\log \mu / \log\log \mu})$ applies to this setting, with any number of processors, where μ is either μ_d or μ_v. The bound holds regardless of whether each job takes the entire capacity of a processor, or if a single job cannot occupy more than a predefined fraction δ of the capacity, for any $\delta > 0$ (see Corollary 5).

In this setting, the bound applies even when the value gained from completing a job is computed in some specific ways: (1) The value of job equals its duration. (2) The value of all jobs is 1. In both cases, the bound holds with respect to μ_d and μ_l, where μ_l is the ratio of largest to smallest load of a job in the input sequence (see Corollaries 6 and 7). Surprisingly, our bound does *not* apply when the value of each job equals its load times its duration. In fact, constant competitive algorithms exist in this case [8, 10, 11, 21, 16, 17]. A lower bound of $\Omega(\sqrt{\mu_v})$ on the competitive ratio of any deterministic preemptive scheduler is shown in [11, 17], where the value of a job is arbitrary. Their bound does not apply to randomized schedulers.

We note that example (b) given at the beginning of the introduction is a generalization of this multiprocessor scheduling problem. Thus our bound applies, yielding similar results.

Next we address **call control** and **virtual circuit routing** problems. Here we have a communication network with guaranteed bandwidth policy in which the links have limited capacities. Requests for connections (or **calls**) arrive through time, where each call has its source and destination nodes, as well as duration, bandwidth requirement, and value. The scheduler must route accepted calls within the capacity limitations of the links; that is, the sum of the bandwidth of calls using each link must be always less than its capacity.

Awerbuch, Azar and Plotkin show a deterministic, non-preemptive $O(\log \mu)$-competitive scheduler in this model, where $\mu = \mu_d \cdot \mu_v \cdot n$ and n is the number of nodes in the network [2]. They also show that this is the best that any deterministic, non-preemptive scheduler can achieve. Their scheduler has the drawback that every job is constrained to require bandwidth at most $1/\log \mu$ of the capacity of any edge. Awerbuch, Bartal *et al.* remove this constraint for networks with a tree topology via a randomized, non-preemptive scheduler [3]. Awerbuch, Gawlick *et al.* improve the bounds for trees and give randomized, non-preemptive schedulers for meshes [4].

Garay and Gopal initiated the study of preemptive call control in [12]. They show constant competitive preemptive schedulers for simple networks and value functions. Garay *et al.* show competitive schedulers on a single link and a line network, for the special case that at most one call can be accommodated on any link, and for several specific ways for determining the value of a call [13]. Bar-Noy *et al.* generalize their results by showing constant competitive schedulers on a single link when the value of a call is the bandwidth times the duration and every call has a bandwidth requirement which is at most a limited fraction of the capacity of the link [8]. Their

strategies apply also to line networks if all calls have infinite durations. (Here the bandwidth times the duration of a call reflects the 'amount of information' contained in the call. Thus, preemption helps when the quantity to be maximized is the throughput of the link.)

Our result provides a lower bound of $\Omega(\sqrt{\log \mu / \log \log \mu})$ on the competitiveness of any preemptive, randomized scheduler for any network, when the value of a call is arbitrary. The bound holds even when a single call cannot occupy more than a predefined fraction δ of the link capacity (for any $\delta > 0$), and if the value of a call is either its duration or 1, as in the multiprocessor scenario described above (see Corollary 8).

Furthermore, if the network has no cycles then our bound applies even when the value of a call equals the duration times bandwidth, or the distance from source to destination, or the distance times duration, or the distance times duration times bandwidth (see Corollary 9). In these cases, $\mu = \min\{\mu_d, D\}$ where D is the diameter of the network.

We suggest the following intuitive description of this state of affairs. The bandwidth times duration measures the "amount of information" contained in a call, whereas the bandwidth times duration times distance measures the "work" invested in a call. It can thus be said that, in the single link case, when the value of a job is directly proportional to the information contained in it then our bound does not apply and constant competitive algorithms exist. If the value of a job is arbitrary then our bound applies. In more complex networks (even in a line of links), our bound applies even when the value of a call is directly proportional to the amount of information, or to the work invested.

Organization. In Section 2 we formally define our generic model. In Section 3 we state and prove the basic lower bound as stated in Theorem 1 (in Section 3.1 we first prove a weaker version of the bound; the proof of this weaker version is simpler and offers intuition for the proof of the full bound). In Section 4 we state and prove several corollaries of our bound. In Section 5 we demonstrate the tightness of the bound by sketching a scheduler in our model, with logarithmic competitiveness.

2 The model

We formalize the generic model described in the Introduction. A **server** is given a sequence of job requests, arriving one by one as time proceeds. We assume that time is discrete, although several requests may arrive at a single time unit. The server can serve at most one job at a time. Each job c is characterized by its arrival time t_c, its duration d_c (known upon arrival), and its **value** v_c. The scheduling of jobs is subject to the following rules. A job has to be either served immediately for the specified duration or rejected. The server can **preempt** (i.e., abort) a job in service. [1] The server accrues an additive gain of v_c for each completed job c. No gain is accrued for preempted jobs.

A sequence $S = c_1, \ldots, c_n$ of job requests is **timely** if $t_{c_i} \geq t_{c_j}$ for every $i > j$. Say that S is **feasible** if no two jobs intersect, that is for no $i \neq j$ we have $t_i < t_j < t_i + d_i$. The **gain** of S is $G(S) \stackrel{\Delta}{=} \sum_{c \in S} v_c$. The **optimal feasible gain** of S is $\mathcal{O}(S) = \max_{\{S' \subseteq S | S' \text{ feasible}\}} G(S')$, where $S' \subseteq S$ means that S' is a subsequence of S.

For a scheduling algorithm $\mathcal{A}$, let $\mathcal{A}(S, r) \subseteq S$ be the sequence of jobs completed by $\mathcal{A}$ on

[1] It may be helpful to visualize each job c as a rectangle with length d_c, height v_c/d_c; the rectangle is located on the number line so that its left edge is at point t_c. The area of the rectangle is v_c.

sequence S and random input r. Algorithm $\mathcal{A}$ is a valid **Scheduler** if whenever S is timely, $\mathcal{A}(S,r)$ is feasible for all r.

Definition 1 *Let $\mu_1(\cdot),\ldots,\mu_t(\cdot)$ be a set of measure functions from request sequences into the reals. A scheduler $\mathcal{A}$ is f-*competitive with respect to *$\mu_1(\cdot),\ldots,\mu_t(\cdot)$ if for all large enough m and for all timely sequences S with $\max_{i=1}^{t}\{\mu_i(S)\} \leq m$ we have:*

$$f(m) \geq \frac{\mathcal{O}(S)}{\mathrm{E}_r(G(\mathcal{A}(S,r)))}$$

where $\mathrm{E}_r(h(r))$ denotes the expected value of $h(r)$ when r is uniformly chosen from the set of random inputs of $\mathcal{A}$. We stress that the sequence S does not depend on the random choices of $\mathcal{A}$ (i.e., using standard terminology, the adversary is oblivious*).*

We sometimes use $\mathrm{E}G\mathcal{A}(S)$ to shorthand $\mathrm{E}_r(G(\mathcal{A}(S,r)))$. We also say that $\frac{\mathcal{O}(S)}{\mathrm{E}G\mathcal{A}(S)}$ is the **competitive ratio of $\mathcal{A}$ on sequence** S. In the sequel we use the following simple observation. For a sequence S, a scheduler $\mathcal{A}$ and a job $c \in S$, let I_c denote the binary random variable having value 1 iff job c is completed in a run of $\mathcal{A}$ on S. Then,

$$\mathrm{E}_r(G(\mathcal{A}(S,r))) = \mathrm{E}_r(\sum_{c\in S} I_c \cdot v_c) = \sum_{c\in S} E_r(I_c)\cdot v_c = \sum_{c\in S} v_c \cdot \mathrm{Prob}(\mathcal{A} \text{ completes } c)$$

That is, the expected gain of the scheduler is the sum over all jobs in the input sequence of the probability that the job is completed times its value.

3 The lower bound

Let $g(x) = \frac{1}{8}\cdot\sqrt{\frac{\log(x)}{\log\log(x)}}$. Let $\mu_d(S)$ (resp., $\mu_v(S)$) be the ratio between the largest and smallest duration (resp., value) of a job in the request sequence S. We prove the following bound:

Theorem 1 *Any randomized, preemptive on-line scheduler is at least g-competitive, with respect to measures μ_d and μ_v.*

For the proof we construct, for each scheduling algorithm $\mathcal{A}$ and infinitely many values m, a timely sequence S with $\max\{\mu_d(S),\mu_v(S)\} \leq m$, and show a feasible "off-line schedule" $S' \subseteq S$ such that $\frac{G(S')}{\mathrm{E}G\mathcal{A}(S)} \geq g(m)$. We first present a very rough description of the construction. Let $S^{(t)}$ denote the prefix of S consisting of the jobs requested up to time t. Say that two jobs are of the same **type** if they have the same duration and value. The sequence S consists of several different types of jobs. Let $p_i(t)$ denote the probability that a job of type i is being served by $\mathcal{A}$ at the end of time unit t. Let $\vec{p}(t) = \langle p_0(t),\ldots,p_k(t)\rangle$, where the number of different types of jobs is $k+1$ Since at most one job can be scheduled at a time we have $\sum_{i=0}^{k} p_i(t) \leq 1$ for all t. Given $\mathcal{A}$, $\vec{p}(t)$ is a function of only $S^{(t)}$.

The construction of the sequence S can be pictured as an interactive game between the scheduler and an adversary, where in each time unit t the adversary generates some requests based on the jobs requested so far and $\vec{p}(t-1)$. Next the scheduler generates $\vec{p}(t)$ based on the new requests and the history of the interaction, subject to the conditions that $\sum_{i=0}^{k} p_i(t) \leq 1$, and $p_i(t) \leq p_i(t-1)$ unless a new i-job is requested at time t. (Here we give the scheduler some extra "leeway" by letting it know in advance all the jobs requested during the entire time unit). Sequence S is now the concatenation of the jobs requested by the adversary in the game.

We stress that S is fixed for each scheduler; it does not depend on the random choices of the scheduler in a specific run.

Our adversary strategy in the above game consists of several recursive applications of roughly the same scheme. In order to better present our construction and analysis, we first describe a simpler adversary that consists of only one application of this scheme. This simpler adversary, called a **1-adversary**, shows a weaker result than Theorem 1, namely that no valid scheduler is less than $(\frac{e+1}{e} - o(1))$-competitive (where e is the base of natural logarithms).

3.1 A 1-adversary

A **1-adversary** generates, for each scheduler $\mathcal{A}$ and a value m, a sequence S with $\mu_d(S) = m^2$ and $\mu_v(S) = m$, and a feasible "off-line schedule" $S' \subseteq S$ such that $\frac{G(S')}{EG\mathcal{A}(S)} \geq \frac{e+1}{e} \cdot \left(1 - O(\frac{1}{m})\right)$.

The 1-adversary uses only two types of jobs: *a*-**jobs** have duration m^2 and value m^2; *b*-**jobs** have duration 1 and value m. The sequence S is constructed as follows. The adversary first requests an a-job at time 0. Next, at each time $t = 0, \ldots, t_0$ (where t_0 is computed below) a b-job is requested. Note that any feasible subsequence of S consists of either the a-job or some of the b-jobs.

Setting an appropriate "stopping time", t_0, is the crux of the adversarial strategy. If the scheduler were deterministic (or alternatively if the adversary could see the random choices of the scheduler), then computing t_0 would be simple: as soon as $\mathcal{A}$ preempts the a-job for some b-job the input sequence would stop. This way, $\mathcal{A}$ gains m while the optimal schedule is the a-job, with gain m^2. If $\mathcal{A}$ never preempts the a-job then we set $t_0 = m^2$ (that is, b-jobs are requested during the entire duration of the a-job), and the optimal schedule is all the b-jobs with gain m^3. In any case, the competitive ratio of $\mathcal{A}$ on S would be m.

However, the random choices of $\mathcal{A}$ are not known. Instead, the adversary will, at the onset of any time unit $t+1$, compute $p_a(t)$, the probability that $\mathcal{A}$ still serves the a-job. If $p_a(t)$ is large enough (i.e., above some threshold described below) then another b-job is requested. Otherwise the request sequence is stopped. The threshold is computed as follows. Let $\mathcal{O}_b(t)$ denote the maximum gain that can be accrued from the requested b-jobs up to time t (that is, $\mathcal{O}_b(t) = t \cdot m$ for time t where $p_a(t)$ has not yet dropped below the threshold). Let the **threshold function** $f(\cdot)$ be:

$$f(x) = \begin{cases} 1 - \alpha e^{\frac{x}{m^2}} & \text{if } x \leq m^2 \\ 1 - \alpha e & \text{if } x > m^2 \end{cases}$$

where $\alpha = \frac{1}{e+1}$. The threshold at time t is $f(\mathcal{O}_b(t))$ (that is, $f(t \cdot m)$). The adversarial strategy can now be described as follows: *First, request an a-job and a b-job at time 0. Next, at each time* $t = 1, \ldots, m^2 - 1$ *do: if* $p_a(t-1) \geq f((t-1)m)$ *request a b-job; else end the request sequence.*

We suggest the following explanation to the choice of threshold function. Let t_0 be the first time that $p_a(t)$ drops below the threshold. It will be seen that the competitive ratio of the algorithm is roughly at most

$$\begin{cases} f(x) + \frac{1}{m}\int_0^x (1 - f(y))dy & \text{if } x \leq m^2 \\ 1 - f(x) & \text{if } x > m^2 \end{cases} \tag{1}$$

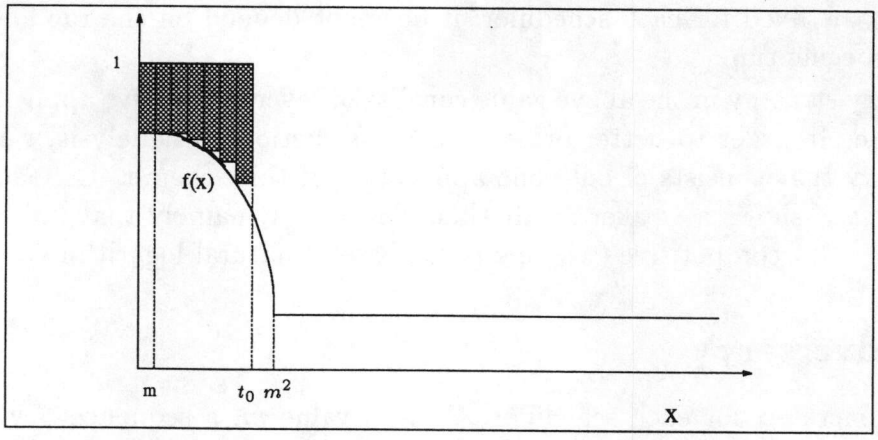

Figure 1: A 1-adversary. The shaded area shows the expected gain of the algorithm from type-b jobs.

where $x = t_0 \cdot m$ and $f(\cdot)$ is the threshold function in use (see Figure 1). Our choice of $f(x)$ ensures that expression (1) evaluates to the same minimal value for all x (that is, for all values of t_0). In particular, for any α the function $f(x) = 1 - \alpha e^{\frac{x}{m^2}}$ solves the differential equation

$$\frac{d}{dx}[f(x) + \frac{1}{m}\int_0^x (1 - f(y))dy] = 0$$

thus making sure that (1) doesn't change for $0 \leq x \leq m^2$. Our choice of α is such that $1 - \alpha = \alpha e$, that is $f(0) = 1 - f(x)$ for all $x > m^2$. Thus (1) has the same value also for $x > m^2$. Consequently, the competitive ratio of $\mathcal{A}$ on S will be roughly the same (minimal) value for all values of t_0.

Analysis of the operation of $\mathcal{A}$ on S. We consider three cases:

Case 1: $t_0 \leq m$. In this case the best feasible subsequence of S is the a-job, with gain m^2 (that is, $\mathcal{O}(S) = m^2$). The expected gain of $\mathcal{A}$ is computed as follows. The probability that $\mathcal{A}$ completes the a-job is $p_a(t_0)$. The probability that $\mathcal{A}$ completes the b-job offered at time t is $p_b(t) \leq 1 - p_a(t)$. Using the observation at the end of Section 2, we have

$$E G\mathcal{A}(S) \leq m^2 p_a(t_0) + m \sum_{t=0}^{t_0} (1 - p_a(t)).$$

It follows from the adversary strategy that $p_a(t_0) < f(t_0 \cdot m)$, and $p_a(t) \geq f(t \cdot m)$ for all $t < t_0$. Thus,

$$\begin{aligned}
E G\mathcal{A}(S) &\leq m^2 f(t_0 \cdot m) + m\left(\sum_{t=0}^{t_0-1}(1 - f(t \cdot m))\right) + m && (2)\\
&\leq m^2 f(t_0 \cdot m) + m\int_0^{t_0}(1 - f(t \cdot m))dx + m && (3)\\
&= m^2(1 - \alpha e^{\frac{t_0}{m}}) + \mathrm{m}\int_0^{t_0} \alpha e^{\frac{x}{m}} dx + \mathrm{m} && (4)\\
&= m^2(1 - \alpha e^{\frac{t_0}{m}}) + \mathrm{m}^2\alpha(e^{\frac{t_0}{m}} - 1) + \mathrm{m} && (5)\\
&= m^2(1 - \alpha) + m && (6)
\end{aligned}$$

(see Figure 1.) Thus, in this case the competitive ratio of $\mathcal{A}$ is at least $\frac{m^2}{m^2(1-\alpha)+m} = \frac{1}{1-\alpha} - O(\frac{1}{m})$.

Case 2: $m < t_0 < m^2 - 1$. In this case the best feasible subsequence of S is all the b-jobs, with gain $t_0 m$. We again have

$$\begin{aligned}
\mathrm{E}G\mathcal{A}(S) &\leq m^2 p_a(t_0) + m\sum_{t=0}^{t_0}(1 - p_a(t)) \\
&\leq m^2(1-\alpha\mathrm{e}) + \mathrm{m}\left(\sum_{\mathrm{t=0}}^{\mathrm{m-1}} \alpha\mathrm{e}^{\frac{\mathrm{t}}{\mathrm{m}}}\right) + \mathrm{m}\left(\sum_{\mathrm{t=m}}^{\mathrm{t_0-1}} \alpha\mathrm{e}\right) + \mathrm{m} \\
&\leq m^2(1-\alpha) + m\left(\sum_{t=m}^{t_0-1} \alpha\mathrm{e}\right) + m \\
&\leq t_0 m\alpha\mathrm{e} + \mathrm{m}
\end{aligned}$$

(the derivation of the third row from the second is done in the same way as in (3) through (3); in the derivation of the fourth row from the third we use the fact that $1 - \alpha = \alpha\mathrm{e}$). Thus, in this case the competitive ratio of $\mathcal{A}$ is at least $\frac{1}{\alpha\mathrm{e}} - O(\frac{1}{m})$.

Case 3: $t_0 = m^2 - 1$. The difference from Case 2 is that here $p_a(t)$ may never go below the threshold. However, it is easy to see that this does not help the scheduler. The best feasible subsequence of S is all the b-jobs, with gain m^3. Bounding the expected gain of the scheduler, we have (since $p_a(t) > 1 - \alpha\mathrm{e}$ for all t):

$$\mathrm{E}G\mathcal{A}(S) \leq m^2 p_a(m^2-1) + m\sum_{t=1}^{m^2-1}(1 - p_a(t)) \leq m^2 + m^3(\alpha\mathrm{e})$$

Thus, the bound on the competitive ratio of $\mathcal{A}$ is the same as in the previous case.

In all three cases, the competitive ratio of $\mathcal{A}$ on S is at least $\min\left\{\frac{1}{1-\alpha}, \frac{1}{\alpha\mathrm{e}}\right\} - O(\frac{1}{m}) = \frac{\mathrm{e}+1}{\mathrm{e}} - O(\frac{1}{m})$. A simple calculation shows that any $m \geq 8$ is enough for showing a non-trivial bound (that is, a bound more than 1).

3.2 Proof of Theorem 1

Roughly speaking, the scheme described in Section 3.1 used the b-jobs to make sure that $\mathcal{A}$ accrues only a fraction of the value of the a-job requested. The same scheme can be used recursively to make sure that $\mathcal{A}$ accrues only a fraction of the value of each b-job requested. That is, use a third type of jobs, called c-jobs, with smaller duration and value than b-jobs. The c-jobs have the property, however, that the total value of a series of consecutive c-jobs scheduled during the duration of a single b-job is greater than the value of the b-job. For each requested b-job recursively apply the above scheme, using the c-jobs. Consequently, the "effective gain" that the scheduler accrues from each b-job is only a fraction of its real value. Thus, a better threshold function can now be used with respect to the a-job, forcing $\mathcal{A}$ to accrue only a smaller fraction of the optimal gain of the entire sequence. The adversary used for proving Theorem 1 uses this idea, implementing several levels of recursion.

We first give a more precise description of the construction and then give some intuition for our choices. Define ADV_k, an adversary implementing k levels of recursion, as follows. ADV_k requests $k+1$ different types of jobs. The value of the jobs of type i (where $i = 0, \ldots, k$) is $v_i = k^{2(k+i)}$ and their duration is $d_i = k^{4i}$. Thus, $\frac{v_{i+1}}{v_i} = k^2$. Using the terms of Section 3.1,

the lower index jobs play the role of the b-jobs and the higher index jobs play the role of the a jobs. Let an i-**period** be the time period in between two consecutive integer multiples of d_i. Let the **partial gain function** $\mathcal{O}_i(t)$ denote the maximum (off-line) gain that can be obtained by scheduling only j-jobs for $j \leq i$ that have been requested by the adversary since the beginning of the current i-period through time t. ADV_k employs k threshold functions $f_1(\cdot), \ldots, f_k(\cdot)$, defined below. As in Section 3.1, the ith threshold at time t is computed by evaluating the ith threshold function at $\mathcal{O}_{i-1}(t)$.

At each time t, the threshold values are compared with the following values, derived from the behavior of the scheduler. Let $q_k(t) = p_k(t)$; for $i \leq k$, let $q_{i-1}(t) = \frac{p_{i-1}(t)}{1-q_i(t)}$. Roughly, the value $q_i(t)$ is at most the probability that an i-job is being served by $\mathcal{A}$ at time t, given that all the j-jobs for $j = i+1, \ldots, k$ were preempted or never scheduled.

ADV_k thus operates as follows. At time $t = 0$, request $k+1$ jobs, one of each type. At each other time $t+1$, run procedure $\text{ADVSTRATEGY}(k, p_k(t))$, described in Figure 2 below.

```
ADVSTRATEGY(i, q_i(t)):
 (1)if d_i divides t then
        Request an i-job;
(2) if i > 0 and f_i(O_{i-1}(t)) ≤ q_i(t) then
        call ADVSTRATEGY(i − 1, p_{i-1}(t)/(1−q_i(t)))
end
```

Figure 2: At each time unit t, ADV_k runs $\text{ADVSTRATEGY}(k, p_k(t))$.

Let:

$$f_i(x) = \begin{cases} 1 - \alpha_i e^{\frac{\beta_{i-1}}{v_i} \cdot x} & \text{if } x \leq v_i \\ 1 - \alpha_i e^{\beta_{i-1}} & \text{if } x > v_i \end{cases}$$

where β_i and α_i are defined as follows. Let $\beta_0 = 1$, and

$$\beta_{i+1} = \beta_i \left[\frac{e^{\beta_i}}{1 + \beta_i \cdot e^{\beta_i}} \right] + \gamma \tag{7}$$

where $\gamma = \frac{2}{k^2}$. Let $\alpha_i = 1 - (\beta_i - \gamma)$. Note that the 1-adversary of Section 3.1 is identical to ADV_1, with $m = k^2$.

We suggest the following explanation to the operation of the adversary. The condition in Step (1) makes sure that the times at which an i-job may be requested are d_i time units apart. Thus, only a single i-job can be requested during an i-period, and the duration of an i-job is a full i-period. The condition in Step (2) of each level i controls whether to call the $(i-1)$st level. Roughly, the $(i-1)$st level is called if the probability that an i-job is being served by $\mathcal{A}$ at time t, given that all the j-jobs for $j = i+1, \ldots k$ were preempted, is more than the threshold. This condition corresponds to the condition of Section 3.1 regarding whether to request any further b-jobs.

Also here, the threshold functions are chosen so that in each level i the scheduler will have the same expected gain regardless of when $q_i(t)$ 'dips' below the threshold. The value $1/\beta_i$ represents the minimum competitive ratio of a scheduler against ADV_i (this statement is formalized in

Lemma 2 below). The competitive ratio of the scheduler against ADV_i (given β_{i-1}) is later shown to be roughly bounded by

$$f_i(x) + \beta_{i-1}v_{i-1}\int_0^x (1 - f_i(y))dy \qquad \text{if } x \leq v_i \tag{8}$$

$$\beta_{i-1}(1 - f_i(x)) \qquad \text{if } x > v_i \tag{9}$$

where $f_i(\cdot)$ is the ith threshold function in use, and $x = \mathcal{O}_{i-1}(t)$. For every α_i and β_i, our choice of $f_i(x)$ ensures that expression (8) evaluates to the same (minimal) value for all $x \leq v_i$, using a differential equation in the same way as in Section 3.1. The α_i's and β_i's are determined as follows. We set $\beta_0 = 1$. Next, for any $i > 0$, α_i is chosen so that the competitive ratios at $x \leq v_i$ and at $x > v_i$ are equal; using (8) and (9) this translates to $f_i(0) = \beta_{i-1}(1 - f_i(v_i))$, or (by the definition of $f_i(x)$):

$$\alpha_i = \frac{1}{\beta_{i-1}e^{\beta_{i-1}} + 1} \tag{10}$$

(for $i = 1$ this condition reduces to the corresponding condition in Section 3.1, that is $\alpha_1 = \frac{1}{e+1}$). Next, β_i can be computed by evaluating (8) at any value of x (and adding a small "error term" γ). Setting, say, $x = 0$ and using (10), we get $\beta_i = f_i(0) + \gamma = 1 - \alpha_i + \gamma = 1 - \frac{1}{\beta_{i-1}e^{\beta_{i-1}}+1} + \gamma$. The recursion relation (7) follows. For the proof we show:

Lemma 2 *Let $k \geq 0$. Let $\mathcal{A}$ be a scheduler and let S be the request sequence generated via an interaction between $\mathcal{A}$ and ADV_k. Then, the competitive ratio of $\mathcal{A}$ on S is at least $\frac{1}{\beta_k+\delta}$, where $\delta = \frac{3}{k-1}$.*

We derive Theorem 1 from Lemma 2. First we claim that $\beta_k = \Theta(\frac{1}{\sqrt{k}})$ (more precisely, $\beta_k + \delta \leq \frac{4}{\sqrt{k}}$). This can be verified from (7) using straightforward algebra. To compute the maximum possible value for k (namely, the maximum possible number of recursion levels), we note that a sequence S generated by ADV_k has $\mu_v(S) = \frac{v_0}{v_k} = k^{2k}$ (and $\mu_d(S) = \frac{v_k}{v_o} = k^{4k}$). Thus, given that $\mu(S) = m$ we can employ ADV_k where $k = \lfloor \frac{\log m}{4 \log\log m} \rfloor$. Theorem 1 follows. ■

Proof of Lemma 2. Consider a sequence S generated via an interaction between $\mathcal{A}$ and ADV_k. We introduce the following notations. A time unit t is called i-**critical** if $\mathcal{O}_i(t) > \mathcal{O}_i(t-1)$. A time interval $\tau = [t_s, t_f]$ is called an i-step if t_s and $t_f + 1$ are i-critical and $t_s + 1, \ldots, t_f$ are not i-critical. For every i, sequence S partitions time into a sequence of i-steps. If $\tau = [t_s, t_f]$ is the first i-step in an i-period then let the i-**height** of τ be $h_i(\tau) = \mathcal{O}_i(t_s)$ (in this case, $\mathcal{O}_i(t_s) = v_i$). Otherwise, let the i-height of τ be $h_i(\tau) = \mathcal{O}_i(t_s) - \mathcal{O}_i(t_s - 1)$. Clearly, the optimal gain from sequence S is the sum of the k-heights of all the k-steps in the duration of S.

Roughly, we will show that the expected gain of $\mathcal{A}$ from the jobs requested during each k-step is at most β_k times the k-height of this step. However, we first distinguish between the main contribution to the competitive ratio of $\mathcal{A}$ and several "special cases" which result in additional, small "error terms". These error terms, encapsuled in the term δ defined above, result from three classes of requests. These classes are defined below and are taken care of in Lemma 4.

A j-job c is said to be **final for level** i if $j < i$ and c is the last j-job to be requested in its i-period. A j-job is **final** if it is final for some level i where $j < i$. A j-job c is a **step-2 job for level** i if $i > j$ and c is requested during the second i-step in its i-period. A job is **step-2** if it is step-2 for some level i with $i > j$. A j-job c is said to be **high-probability** if $q_j(t) \geq f_j(\mathcal{O}_{j-1}(t))$

for all times t in its duration (high probability jobs correspond, in principle, to Case 3 in the analysis of a 1-adversary in Section 3.1 where the probability that a job is running never falls below the threshold). A job is **regular** if it is neither a step-2, final or high-probability job.

Let $EG\mathcal{A}_\tau(S)$ denote the expected gain of $\mathcal{A}$ from the *regular* jobs requested by ADV_k during some step τ (that is, $EG\mathcal{A}_\tau(S) = \sum_{c \in S_\tau} v_c \cdot \text{Prob}(\mathcal{A} \text{ completes } c)$, where S_τ is the sequence of regular jobs requested during τ). Lemma 3 states that $EG\mathcal{A}_\tau(S)$ is at most β_k times the k-height of each k-step τ. We know that $\mathcal{O}(S) = \sum_{\{k\text{-steps } \tau\}} h_k(\tau)$. It follows that the expected gain of $\mathcal{A}$ from S due to regular jobs is at most $\beta_k \cdot \mathcal{O}(S)$. Lemma 4 states that the total gain of $\mathcal{A}$ from all the non-regular jobs in S is at most $\delta \cdot \mathcal{O}(S)$. Lemma 2 follows. ∎

In the rest of the proof we use the following observations regarding the structure of S. A job is said to be i-**small** if it is a j-job with $j < i$. The first job requested in each i-period is an i-job. The rest of the jobs in this i-period are i-small. Each i-period can be partitioned into i-steps (any i-step is contained within a single i-period). The first i-step in an i-period occurs due to the request of the i-job and has i-height v_i. This i-step can be partitioned in several $(i-1)$-steps. In the second i-step (if there is one), the optimal gain due to the i-small jobs exceeds the gain of the i-job for the first time. If this happens, then the optimal schedule in the i-period contains only i-small jobs instead of the i-job. As a result, any additional increase in $\mathcal{O}_i$ within this i-period causes the same increase in $\mathcal{O}_{i-1}$, and any subsequent i-step τ is also an $(i-1)$-step. Furthermore, $h_i(\tau) = h_{i-1}(\tau)$. Note that if there is a second i-step τ, then τ is also an $(i-1)$-step but it is not necessarily the case that $h_i(\tau) = h_{i-1}(\tau)$. This is because the arrival of an i-small job may cause the optimal gain from i-small jobs to exceed the value of the i-job by only a small amount.

Lemma 3 *Let $k \geq 0$. Let $\mathcal{A}$ be a scheduler and let S be the request sequence generated via an interaction between $\mathcal{A}$ and ADV_k. Let τ be a k-step. Then,*

$$EG\mathcal{A}_\tau(S) \leq \beta_k \cdot h_k(\tau) \tag{11}$$

Proof. We prove the lemma by induction on k. The lemma trivially holds for ADV_0 (using one type of job), since $\beta_0 = 1$. Also, the case of ADV_1 was analyzed in Section 3.1.

Let $k > 0$ and assume that there exists a scheduler $\mathcal{A}$ and a k-step τ such that (11) is violated. We construct a scheduler $\mathcal{A}'$ with the following property that contradicts the induction hypothesis. Consider the sequence S' generated via the interaction of $\mathcal{A}'$ with ADV_{k-1}. Then there exists a $(k-1)$-step τ' in S' such that

$$EG\mathcal{A}'_{\tau'}(S') > \beta_{k-1} \cdot h_{k-1}(\tau') \tag{12}$$

(where $EG\mathcal{A}'_{\tau'}(S')$ is defined similarly to $EG\mathcal{A}_\tau(S)$, with respect to $\mathcal{A}'$, S', and τ').

Scheduler $\mathcal{A}'$ is constructed as follows. Consider the sequence S generated by ADV_k interacting with $\mathcal{A}$. A $(k-1)$-period in S, starting at time t_*, is chosen in a way described below. Roughly, $\mathcal{A}'$ will imitate the operation of $\mathcal{A}$ starting at time t_*, conditioned on the event that $\mathcal{A}$ has preempted the k-job. More precisely, let $p_i(t)$ (resp., $p'_i(t)$) denote the probability that $\mathcal{A}$ (resp., $\mathcal{A}'$) has a job of type i scheduled at time unit t. Scheduler $\mathcal{A}'$ is constructed so that when playing against ADV_{k-1} for the duration of a $(k-1)$-period, the following probability vector $\vec{p}'(t) = p'_0(t), \ldots, p'_{k-1}(t)$ is maintained:

$$p'_i(t) = \frac{p_i(t+t_*)}{1 - p_k(t+t_*)}. \tag{13}$$

Scheduler $\mathcal{A}'$ can always be implemented since $\sum_{i=0}^{k-1} p_i'(t) \leq 1$, and $p_i'(t) > p_i'(t-1)$ only if an i-job is requested at time t. In choosing t_* we distinguish between two cases.

Case 1: step τ is not the first k-step. Then τ is also a $(k-1)$-step, and is contained in a $(k-1)$-period. Let t_* be the beginning of this $(k-1)$-period.

Case 2: step τ is the first k-step. The first k-step is partitioned into several $(k-1)$-steps, $\tau_1, \ldots, \tau_l$. Let τ_* be the $(k-1)$-step contained in τ that maximizes the ratio

$$\max_{1 \leq i \leq l} \left\{ \frac{\mathrm{E}G\mathcal{A}_{\tau_i}(S)}{(1 - p_k(e_{\tau_i})) \cdot h_{k-1}(\tau_i)} \right\}$$

among all the $(k-1)$-steps contained in τ, where e_{τ_i} is the latest ending time of a regular job requested during τ_i (note that e_{τ_i} may be later than the end of τ_i). For convenience, we assume that the initial k-job is not included in the first $(k-1)$-step. Let t_* be the beginning of the $(k-1)$-period containing τ_*. Note that $\mathcal{A}'$ may perform very poorly on request sequences different than those generated by ADV_{k-1}. The only purpose of $\mathcal{A}'$ is to contradict the induction hypothesis.

Analysis of scheduler $\mathcal{A}'$: It can be seen from (13) and the adversary strategy (Figure 2) that at each time t ADV_{k-1}, interacting with $\mathcal{A}'$, requests exactly the same jobs that ADV_k requests at time $t + t_*$ when interacting with $\mathcal{A}$; this holds with the following two exceptions. First, ADV_k may request an initial k-job at time t_*, where ADV_{k-1} does not. Second, the sequence S may end before S', if $p_k(t)$ falls below the threshold for some time t. Let $\hat{S}$ denote the sequence S starting at time t_* with the initial k-job removed (if there is one). Then, $\hat{S}$ is a prefix of S'. Furthermore, for any $(k-1)$-step τ in $\hat{S}$ we have $h_{k-1}(\tau) = h_{k-1}(\tau')$, where τ' is the $(k-1)$-step in S' that corresponds to τ. Next we distinguish three cases:

Case 1: The induction hypothesis is violated with respect to the first k-step, τ. Let τ_* be as in the definition of $\mathcal{A}'$. Let τ_*' be the $(k-1)$-step that corresponds to τ_* with respect to S'. We show that if $\mathrm{E}G\mathcal{A}'_{\tau_*'}(S') \leq \beta_{k-1} \cdot h_{k-1}(\tau_*')$ then $\mathrm{E}G\mathcal{A}_{\tau}(S) \leq \beta_k \cdot v_k$.

We first express $\mathrm{E}G\mathcal{A}_{\tau}(S)$ in terms of the expected gain of $\mathcal{A}$ in the $(k-1)$-steps $\tau_1, \ldots, \tau_l$ contained in τ. Assume that the initial k-job is not a high-probability job. In this case, there is a time t such that $p_k(t) < f_k(\mathcal{O}_{k-1}(t))$. Let $t_0 = t$ if t happens before the end of τ; otherwise t_0 is the first time unit after τ. We can bound the expected cost of $\mathcal{A}$ in τ as follows,

$$\mathrm{E}G\mathcal{A}_{\tau}(S) \leq v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \sum_{i=1}^{l} \mathrm{E}G\mathcal{A}_{\tau_i}(S). \tag{14}$$

(Inequality (14) is satisfied even if the initial k-job a is a high-probability job. In this case, the contribution of a is not included in $\mathrm{E}G\mathcal{A}_{\tau}(S)$; it is considered in Lemma 4.)

Scheduler $\mathcal{A}'$ completes each job $c' \in S'$ requested during τ_* with probability at least $(1 - p_k(e_c))$ times the probability that $\mathcal{A}$ completes the corresponding job $c \in S$. Since $e_c \leq e_{\tau_*}$, $\mathrm{E}G\mathcal{A}_{\tau_*}(S) \leq (1 - p_k(e_{\tau_*})) \cdot \mathrm{E}G\mathcal{A}'_{\tau_*'}(S')$. Now, assume that $\mathrm{E}G\mathcal{A}'_{\tau_*'}(S') \leq \beta_{k-1} \cdot h_{k-1}(\tau_*')$, and recall that $h_{k-1}(\tau_*') = h_{k-1}(\tau_*)$. Thus,

$$\mathrm{E}G\mathcal{A}_{\tau_*}(S) \leq (1 - p_k(e_{\tau_*})) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_*).$$

It follows from the choice of τ_* that $\mathrm{E}G\mathcal{A}_{\tau_i}(S) \leq (1 - p_k(e_{\tau_i}))\beta_{k-1} \cdot h_{k-1}(\tau_i)$ for all steps τ_i. Thus, (14) implies that

$$\mathrm{E}G\mathcal{A}_{\tau}(S) \leq v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \sum_{i=1}^{l} (1 - p_k(e_{\tau_i})) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_i).$$

We complete the analysis of this case by showing that

$$v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \sum_{i=1}^{l}(1 - p_k(e_{\tau_i})) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_i) \;\leq\; \beta_k \cdot v_k.$$

First, we show that

$$v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \sum_{i=1}^{l}(1 - p_k(s_{\tau_i})) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_i) \leq (\beta_k - \gamma) \cdot v_k \tag{15}$$

where s_{τ_i} is the starting time of step τ_i, and $\gamma = \frac{2}{k^2}$ is a small "error term". Next we show that

$$\sum_{i=1}^{l}(p_k(s_{\tau_i}) - p_k(e_{\tau_i})) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_i) \leq \gamma \cdot v_k. \tag{16}$$

Showing (15), we have

$$\begin{aligned}
& v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \sum_{i=1}^{l}(1 - p_k(s_{\tau_i})) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_i) \\
\leq\; & v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \sum_{i=1}^{l}(1 - f_k(\mathcal{O}_{k-1}(s_{\tau_i}))) \cdot \beta_{k-1} \cdot h_{k-1}(\tau_i) \\
\leq\; & v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \beta_{k-1} \cdot \int_0^{\mathcal{O}_{k-1}(t_0)}(1 - f_k(y))dy
\end{aligned}$$

We integrate only up to $y = \mathcal{O}_{k-1}(t_0)$, since either t_0 is after the end of τ or no non-final jobs are requested at or after time t_0. Let $x = \mathcal{O}_{k-1}(t_0)$. Since this is the first k-step we have $x \leq v_k$; thus, $f_k(x) = 1 - \alpha_k e^{\frac{\beta_{k-1}x}{v_k}}$. Thus:

$$\begin{aligned}
& v_k \cdot f_k(\mathcal{O}_{k-1}(t_0)) + \beta_{k-1} \cdot \int_0^{\mathcal{O}_{k-1}(t_0)}(1 - f_k(y))dy \\
=\; & (1 - \alpha_k e^{\frac{\beta_{k-1}x}{v_k}})v_k + \beta_{k-1}\int_0^x \alpha_k e^{\frac{\beta_{k-1}y}{v_k}}dy \\
=\; & (1 - \alpha_k e^{\frac{\beta_{k-1}x}{v_k}})v_k + \beta_{k-1}\frac{v_k}{\beta_{k-1}}\alpha_k(e^{\frac{\beta_{k-1}x}{v_k}} - 1) \\
=\; & (1 - \alpha_k)v_k \\
=\; & (\beta_k - \gamma)v_k.
\end{aligned}$$

Next, we show (16). Let w_i denote the largest j such that a j-job arrives on the first time unit of τ_i. Then $h_{k-1}(\tau_i) \leq v_{w_i} = k^{2(k+w_i)}$. Rewriting the left hand side of (16) we get

$$\beta_{k-1}\sum_{i=1}^{l} h_{k-1}(\tau_i) \cdot (p_k(s_{\tau_i}) - p_k(e_{\tau_i})) \leq \beta_{k-1}\sum_{j=0}^{k-1} v_j \cdot \sum_{\{i\,:\,w_i=j\}}(p_k(s_{\tau_i}) - p_k(e_{\tau_i}))$$

Observe that if $w_i = w_{i'}$ (and $i > i'$) then step τ_i begins only after all the jobs requested during $\tau_{i'}$ have ended (this is so since at most one j-job is requested in a j-period). Since $p_k(t)$ is a non-increasing function, we have:

$$\sum_{j=0}^{k-1} v_j \cdot \sum_{\{i\,\mid\,w_i=j\}}(p_k(s_{\tau_i}) - p_k(e_{\tau_i})) \leq \sum_{j=0}^{k-1} v_j \cdot 1 \leq 2 \cdot v_{k-1} \leq \frac{2}{k^2} \cdot v_k = \gamma \cdot v_k$$

Case 2: Step τ is the second k-step. All jobs requested in this step are step-2 jobs and are thus not regular. Thus, $EG\mathcal{A}_\tau(S) = 0$. (Step-2 jobs are accounted for in Lemma 4.)

Case 3: Step τ is neither the first nor the second k-step. Then, τ is a $(k-1)$-step as well. Let τ' be the $(k-1)$-step that corresponds to τ in S'. Then, $h_k(\tau) = h_{k-1}(\tau) = h_{k-1}(\tau')$. We show below that the probability that $\mathcal{A}$ completes each regular job $c \in S$ is at most $\frac{\beta_k}{\beta_{k-1}}$ times the probability that $\mathcal{A}'$ completes the corresponding job $c' \in S'$ during τ'. Thus, (12) holds with respect to step τ'.

Scheduler $\mathcal{A}$ completes an i-job c with probability $p_i(e_c)$, and $\mathcal{A}'$ completes c' with probability $\frac{p_i(e_c)}{1-p_k(e_c)}$, where e_c is the time at which c ends (i.e. $e_c = t_c + d_c$). It remains to show that $1 - p_k(e_c) \leq \frac{\beta_k}{\beta_{k-1}}$. Since c is not a final job, we have that $p_k(e_c) \geq f_k(\mathcal{O}(e_c))$. Since τ is not the first k-step, we have that $\mathcal{O}_{k-1}(e_c) > v_k$. Thus, $f_k(\mathcal{O}_{k-1}(e_c)) = 1 - \alpha_k e^{\beta_{k-1}}$, and $1 - p_k(e_c) \leq \alpha_k e^{\beta_{k-1}} \leq \frac{\beta_k}{\beta_{k-1}}$. ■

Lemma 4 *Let $k \geq 0$. Let $\mathcal{A}$ be a scheduler and let S be the request sequence generated by an interaction between $\mathcal{A}$ and ADV_k. Then, the total gain of $\mathcal{A}$ from all the step-2, final and high-probability jobs in S is at most $\frac{3}{k-1}\mathcal{O}(S)$.*

Proof. We first bound the gain of $\mathcal{A}$ from the final and step-2 jobs. The gain achievable by the optimal algorithm (which upper bounds the gain achievable by $\mathcal{A}$) from all the step-2 jobs for level i requested within a given i-period is at most v_{i-1}. To see this, suppose that an $(i-1)$-step begins at time t and j is the largest number such that a j-job arrives at time t. It must be the case that $j \leq i-1$. A new $(i-1)$-step (and hence also i-step) begins when the optimal gain achievable from j-small calls exceeds v_j. Thus the optimal gain in the second step is at most $v_j \leq v_{i-1}$.

We can bound the optimal gain achievable from final jobs by the total value of all final jobs. Notice that for each i-job and every $j < i$, there is at most one final j-job. Let n_i be the total number of i-jobs in S. The optimal gain achievable from the jobs that are final or step-2 for level i is thus at most

$$n_i(v_{i-1} + \sum_{j=0}^{i-1} v_j) \leq 2n_i v_i \sum_{j=0}^{i-1} \frac{1}{k^{2(i-j)}} \leq \frac{2n_i v_i}{k^2 - 1}$$

For every i, the sequence of all i-jobs in S is a feasible subsequence of S, with gain $n_i v_i$. Therefore, $\mathcal{O}(S) \geq \max_i(n_i v_i)$. The sum of the values of all final and step-2 jobs is thus at most:

$$\sum_{i=0}^{k} \frac{2n_i v_i}{k^2 - 1} \leq (k+1) \max_i \frac{2n_i v_i}{k^2 - 1} \leq \frac{2}{k-1}\mathcal{O}(S).$$

Next we bound the gain achievable from the high-probability jobs which are not final. Consider a high-probability i-job c which is not final. We first show that during every $(i-1)$ period within the duration of c, an $(i-1)$-job is requested. To see why this is true, recall that an $(i-1)$ job is requested in an $(i-1)$-period if at the first time unit t in the period $q_j(t) < f_j(\mathcal{O}_{j-1}(t))$ for all $j \geq i$. If c is not final, then $q_j(t) < f_j(\mathcal{O}_{j-1}(t))$ for all $j > i$ and for all times t in the duration of c. If c is high-probability then $q_i(t) < f_i(\mathcal{O}_{i-1}(t)$ for all times t in the duration of c.

Consequently, k^4 jobs of type $(i-1)$ are requested in the duration of c. The total value of these $(i-1)$-jobs is k^2 times the value of c. Let h_i be the number of all non-final high-probability i-jobs in S. We know that $k^2 v_i h_i \leq v_{i-1} n_{i-1}$. Therefore, the total gain from all

non-final high-probability jobs is at most

$$\sum_{i=1}^{k} v_i h_i \leq \frac{1}{k^2} \sum_{i=1}^{k} v_{i-1} n_{i-1} \leq \frac{1}{k} \max_i v_i n_i \leq \frac{1}{k} \mathcal{O}(S). \quad \blacksquare$$

4 Applications of the bound

In this section, we present a series of corollaries showing how the lower bound of Theorem 1 can be generalized and extended to related problems.

We first consider the multiprocessor scheduling problem described in Section 1.1. Below we describe how the bound in Theorem 1 can be generalized to situations where more than a single job can be served at a time. Each job c has a load l_c, and a feasible sequence is one in which the sum of the loads of any set of overlapping jobs is at most U. We also show that the bound holds if there is more than one server (each with capacity U) and if there is an upper bound on the load that any single job may require.

Recall that μ_d, μ_l and μ_v denote the ratio of maximum to minimum duration, load and value of a job, respectively. Let $g(x) = \frac{1}{8}\sqrt{\frac{\log x}{\log\log x}}$.

Corollary 5 *Theorem 1 holds even when there are m servers, each with capacity U, and for every job c, $l_c \leq \delta U$ for any $\delta \leq 1$.*

Proof. We assume the existence of a scheduler $\mathcal{A}'$ that contradicts the corollary. Based on $\mathcal{A}'$, we then construct scheduler $\mathcal{A}$ that contradicts Theorem 1. $\mathcal{A}$ keeps a copy of $\mathcal{A}'$ running. Given a sequence S of requests, $\mathcal{A}$ gives $\mathcal{A}'$ a sequence S' constructed as follows.

Let α be chosen so that $1/\alpha = \lceil 1/\delta \rceil$. Upon receiving a request c in S, $\mathcal{A}$ generates a set S'_c of $N_c = m/\alpha$ requests for $\mathcal{A}'$. Each job in S'_c has the same duration as c, has value $v_{c'} = \alpha v_c/m$ and load $l_{c'} = U\alpha$. Note that if all jobs from S'_c are scheduled, they use up the entire capacity of the system. Furthermore, the sum of the values of all the jobs in S'_c is equal to v_c (i.e. $N_c v_{c'} = v_c$).

First observe that $\mathcal{O}(S) \leq \mathcal{O}(S')$. Given a feasible subsequence $\hat{S} \subseteq S$, we can obtain a feasible subsequence $\hat{S}' \subseteq S'$ such that $G(\hat{S}) \leq G(\hat{S}')$: for every $c \in \hat{S}$, include all the jobs from S'_c in $\hat{S}'$.

Next we describe the behavior of $\mathcal{A}$. Let $n_c(t)$ denote the expected number of jobs in S'_c scheduled by $\mathcal{A}'$ at time t. $\mathcal{A}$ will accept and preempt jobs so as to maintain the invariant that at all times t, the probability that $\mathcal{A}$ has job c scheduled is $n_c(t)/N_c$. In showing how $\mathcal{A}$ achieves this we assume that $\mathcal{A}'$ has the property that after the arrival of a set of jobs S'_c, the entire capacity of all m servers is being used. No generality is lost by this assumption, since $\mathcal{A}$ can always decide to preempt jobs later in favor of an incoming job. Now suppose that $\mathcal{A}$ has a job $\tilde{c}$ running during time unit $t-1$ and jobs $c_1, c_2, \ldots, c_k$ arrive at time t. $\mathcal{A}$ will preempt $\tilde{c}$ with probability $(n_{\tilde{c}}(t-1) - n_{\tilde{c}}(t))/n_{\tilde{c}}(t-1)$. If $\mathcal{A}$ preempts its current job or did not have any jobs running during time $t-1$, then it accepts job c_i with probability $n_{c_i}(t)/\sum_{j=1}^{k} n_{c_j}(t)$. It is easy to verify by induction on t that the invariant holds.

Let e_c be the ending time of job c. The expected gain of $\mathcal{A}'$ from the jobs in S'_c is $v_{c'} n_c(e_c) = v_c n_c(e_c)\alpha/m$. The expected gain of $\mathcal{A}$ from job c is $v_c n_c(e_c)/N_c = v_c n_c(e_c)\alpha/m$. Thus, the expected gain of $\mathcal{A}$ on its input sequence S is equal to the expected gain of $\mathcal{A}'$ on the constructed sequence S'.

Since $\mu_d(S) = \mu_d(S')$ and $\mu_v(S) = \mu_v(S')$, the existence of $\mathcal{A}'$ contradicts Theorem 1. ■

Each of the remaining corollaries is proven using a reduction technique similar to the proof of Corollary 5, We first describe this technique in more general terms. Next, we state the corollaries and fill out the necessary details in the proofs.

Assume the existence of a scheduler $\mathcal{A}'$ that contradicts some corollary. Based on $\mathcal{A}'$, we then construct scheduler $\mathcal{A}$ that contradicts Theorem 1. $\mathcal{A}$ keeps a copy of $\mathcal{A}'$ running. Given a sequence S of requests, $\mathcal{A}$ gives $\mathcal{A}'$ a sequence S' constructed as follows. Upon receiving a request c in S, $\mathcal{A}$ generates a set S'_c of N_c requests for $\mathcal{A}'$ each of value $v_{c'}$ and load $l_{c'}$. The duration of each job in S'_c is exactly d_c. The parameters of the constructed requests depend on the reduction. However, they will always have the property that the collection of the N_c jobs in S'_c, if all scheduled, use all of the capacity of the system. That is, $l_{c'}N_c$ is the total capacity of all servers. Furthermore, the sum of the values of all the jobs in S'_c is equal to v_c (i.e. $N_c v_{c'} = v_c$).

Given these properties of the reduction, we can deduce that $\mathcal{O}(S) \leq \mathcal{O}(S')$. Given a feasible subsequence $\hat{S} \subseteq S$, we can obtain a feasible subsequence $\hat{S}' \subseteq S'$ such that $G(\hat{S}) \leq G(\hat{S}')$: for every $c \in \hat{S}$, include all the jobs from S'_c in $\hat{S}'$.

Let $n_c(t)$ denote the expected number of jobs in S'_c scheduled by $\mathcal{A}'$ at time t. Let e_c be the ending time of job c. Then, the expected gain of $\mathcal{A}'$ from the jobs in S'_c is $v_{c'}n_c(e_c)$. As above, $\mathcal{A}$ maintains the invariant that at all times t the probability that $\mathcal{A}$ has job c scheduled is $n_c(t)/N_c$. Consequently, the expected gain of $\mathcal{A}$ from job c is $v_c n_c(e_c)/N_c$.

The next two corollaries show that the bound applies also when some specific value functions are used.

Corollary 6 *Assume jobs have arbitrary load and duration, and the value of a job is its duration. Then, any randomized, preemptive on-line scheduler is at least $g/2$-competitive for measures μ_d and μ_l.*

Proof. We choose $N_c = v_c/d_c$, $l_{c'} = \frac{Ud_c}{v_c}$, and $d_{c'} = d_c$. The expected gain of $\mathcal{A}'$ from jobs in S'_c is $d_c n_c(e_c)$, and the expected gain of $\mathcal{A}$ from job c is $v_c n_c(e_c)/N_c = d_c n_c(e_c)$. Thus, the expected gain of $\mathcal{A}$ on its input sequence S is equal to the expected gain of $\mathcal{A}'$ on the constructed sequence S'. Since $\mu_d(S) = \mu_d(S')$ and $\mu_l(S') \leq \mu_d(S) \cdot \mu_v(S)$, if $\mathcal{A}'$ is $g/2$-competitive with respect to measures μ_d and μ_l, then $\mathcal{A}$ is g-competitive with respect to measures μ_d and μ_v, thus contradicting Theorem 1.

We remark that the construction can be easily adapted to the cases where more than a single job can be served at a time, where there is a bound δU (for any $\delta \leq 1$) on the load of a single job and where there are m servers. This is accomplished in a similar manner as in Corollary 5, that is by increasing the number of jobs by a factor of m/δ and reducing the load of each job by a factor of δ. ■

Corollary 7 *Assume jobs have arbitrary load and duration, and the value of all jobs is 1. Then, any randomized, preemptive on-line scheduler is at least g-competitive for measures μ_l and μ_d.*

Proof. We choose $N_c = v_c$, $l_{c'} = \frac{U}{v_c}$, and $d_{c'} = d_c$. The expected gain of $\mathcal{A}'$ from the S'_c jobs is $n_c(e_c)$, and the expected gain of $\mathcal{A}$ from job c is $v_c n_c(e_c)/N_c = n_c(e_c)$. Thus, the expected gain of $\mathcal{A}$ on its input sequence S is equal to the expected gain of $\mathcal{A}'$ on the constructed sequence S'. Since $\mu_d(S) = \mu_d(S')$ and $\mu_l(S') = \mu_v(S)$, the existence of $\mathcal{A}'$ contradicts Theorem 1.

Again, the construction can be easily adapted as in as in Corollary 5 to the cases where more than a single job can be served at a time, where there is a bound δU (for any $\delta \leq 1$) on the load of a single job and where there are m servers. ■

We note that our bound does *not* hold in the scenario where the value of a job is its duration times its load. In fact, using this value function there exist constant-competitive schedulers [8].

Next we address the call control model, described in Section 1.1. The job scheduling for the uniprocessor model discussed above is identical to the call control model on a single link (when load is translated to bandwidth). We can generalize the result to work for an arbitrary network as follows:

Corollary 8 *Assume calls have arbitrary values. Then, any randomized, preemptive call control algorithm for any network has a competitive ratio of at least g for measures μ_d and μ_v.*

Proof. Pick any pair of nodes s and t in the network. Let F be the value of the maximum flow from s to t where $u(e)$ is the capacity of an edge. Let ϵ be the greatest common divisor of all the edge capacities. We will only request calls between s and t with bandwidth at most ϵ. Thus, the set of paths between s and t can be treated as a single edge with capacity F.

We use the same reduction above, choosing $N_c = F/\epsilon$, $v_{c'} = v_c\epsilon/F$, $d_{c'} = d_c$ and $l_c = \epsilon$. The expected gain of $\mathcal{A}'$ from the c' jobs is $v_c n_c(e_c)\epsilon/F$, and the expected gain of $\mathcal{A}$ from job c is $v_c n_c(e_c)/N_c = v_c n_c(e_c)\epsilon/F$. Thus, the expected gain of $\mathcal{A}$ on its input sequence S is equal to the expected gain of $\mathcal{A}'$ on the constructed sequence S'. Since $\mu_d(S) = \mu_d(S')$ and $\mu_v(S) = \mu_v(S')$, the existence of $\mathcal{A}'$ contradicts Theorem 1. ■

The techniques of Corollary 8 can easily be extended to show that the bounds of Corollaries 6 and 7 (using $v_c = d_c$ or $v_c = 1$) apply to the corresponding call control problems in any network topology. Furthermore, when the network has no cycles we can extend the lower bound to special cases where the value of a call is determined in some specific ways, listed below. Let r_c denote the distance between the two endpoints of the call c.

Corollary 9 *Suppose we have a tree network with diameter D. Then any randomized, preemptive call control algorithm for this network has a competitive ratio of at least $\Omega(g)$ for measures μ_d and D, even with the following criteria for call value:*

(i) $v_c = l_c d_c$.
(ii) $v_c = r_c$.
(iii) $v_c = r_c d_c$.
(iv) $v_c = r_c l_c d_c$.

Remarks: (1) Note that although D is not a parameter of the input sequence, it describes the network which along with the sequence is part of the input to the problem. The definition of competitiveness in Section 2 can be adapted accordingly. (2) Method (i) for determining the value of a call measures the amount of information potentially contained in a call. Measure (iv) measures the amount of "work" invested in a call. The fact that our bound holds for these measures stands in contrast to the single-edge and multiprocessor cases. There, constant competitive algorithms exist for similar measures.

Proof. We employ the usual reduction to the setting in Theorem 1.

(i) Assume the existence of a $o(g)$-competitive scheduler $\mathcal{A}'$ on a tree network with diameter $D = \mu_d(S) \cdot \mu_v(S)$, where the value of a call c is $l_c \cdot d_c$. We construct a scheduler $\mathcal{A}$ that contradicts Theorem 1.

In our construction we assume that each job c given to $\mathcal{A}$ has the property that v_c/d_c divides the diameter D. Such a situation can be easily acheived by normalizing the value of each job so that $\min_c\{v_c/d_c\} = 1$. The value of each job c can then be rounded so that v_c/d_c is a power of 2, and the diameter of the network is rounded down to the nearest power of two. It is easy to see that this way only a factor of two is lost in the effectiveness of the reduction. Note that Theorem 1 holds even when the algorithm knows minimum value of v_c/d_c in the sequence in advance.

$\mathcal{A}$ first chooses a simple path of length D in the network on which $\mathcal{A}'$ operates and numbers the nodes along the path $0, 1, \ldots, D$. For each requested job c, scheduler $\mathcal{A}$ requests a set S'_c of v_c/d_c calls for $\mathcal{A}'$, where the ith call starts at node $\left\lceil (i-1) \cdot \frac{Dd_c}{v_c} \right\rceil$ and ends at node $\left\lceil i \cdot \frac{Dd_c}{v_c} \right\rceil$. (That is, $N_c = v_c/d_c$ and each generated call c' is of length $r_{c'} = D\frac{d_c}{v_c}$.) All generated calls have $l_{c'} = 1$. The expected gain of $\mathcal{A}'$ from the jobs in S'_c is $d_c n_c(e_c)$, and the expected gain of $\mathcal{A}$ from job c is $v_c n_c(e_c) d_c/v_c = d_c n_c(e_c)$. Thus, the expected gain of $\mathcal{A}$ on its input sequence S is equal to the expected gain of $\mathcal{A}'$ on the constructed sequence S'. Since $\mu_d(S) = \mu_d(S')$ and $\mu_v(S) \cdot \mu_d(S) \geq D$, the existence of $\mathcal{A}'$ contradicts Theorem 1.

(ii) In cases (ii) through (iv) we assume that for each job c requested of $\mathcal{A}$, the values v_c, and v_c/d_c are integers. Also here, such a situation can be acheived by rounding, with a loss of at most a factor of two in effectiveness.

As in the proof of (i), $\mathcal{A}$ picks a path of length D in the tree. The vertices of this path are labelled $\{0, \ldots, D\}$. $N_c = 1$, $l_c = 1$, $d_c = d_{c'}$. The endpoints of the call in S'_c are node 0 and node v_c. Thus, $v_c = r_c = v_{c'}$. Since $\mu_d(S) = \mu_d(S')$ and $\mu_v(S) \geq D$, the existence of $\mathcal{A}'$ contradicts Theorem 1.

(iii) As in the proof of (i), $\mathcal{A}$ picks a path of length D in the tree. The vertices of this path are labelled $\{0, \ldots, D\}$. $N_c = 1$, $l_c = 1$, $d_c = d_{c'}$. The endpoints of the call in S'_c are 0 and node v_c/d_c. Thus, $r_{c'} = v_c/d_c$, and $v_c = r_c d_{c'} = v_{c'}$. Since $\mu_d(S) = \mu_d(S')$ and $\mu_v(S) \cdot \mu_d(S) \geq D$, the existence of $\mathcal{A}'$ contradicts Theorem 1.

(iv) The construction for (iii) works here since for all c, $l_c = 1$.

5 Tightness of the bound

We sketch two simple preemptive, randomized schedulers in the basic model defined in Section 2. The first is $O(\log \mu_v)$-competitive and the second is $O(\log \mu_d)$-competitive. Depending on whether $\mu_d \leq \mu_v$, we can implement the appropriate one to obtain a $O(\log \mu)$-competitive scheduler for $\mu = \min\{\mu_v, \mu_d\}$.

For the $O(\log \mu_v)$-competitive scheduler, we use a technique of [3]. Let m (resp., M) be the minimum (resp., maximum) possible value of a job. Let $k = \lceil \log(M/m) \rceil$ and let $v_0, \ldots, v_k$ satisfy $v_0 = m$, $v_k = M$, and $v_i/v_{i-1} \leq 2$. Our scheduler first picks $1 \leq i \leq k$ at random. Next, it rejects all jobs whose value does not fall between v_{i-1} and v_i. Jobs with value between v_{i-1} and v_i are scheduled as follows: accept an incoming job if it terminates before the currently running job. It is easy to see that this simple scheduler is 2-competitive over the sequence of jobs in the chosen range. Thus, the competitive ratio of the randomized algorithm is $4 \log_2 \mu_p = 4 \log_2(M/m)$.

For the $O(\log \mu_d)$-competitive scheduler, let d (resp., D) be the minimum (resp., maximum) possible duration of a job. Let $k = \lceil \log(D/d) \rceil$ and let $d_0, \ldots, d_k$ satisfy $d_0 = m$, $d_k = M$, and $d_i/d_{i-1} \leq 2$. Our scheduler first picks $1 \leq i \leq k$ at random. Next, it rejects all jobs whose duration does not fall between d_{i-1} and d_i. Jobs with duration between d_{i-1} and d_i are scheduled as follows: accept an incoming job if there is no currently running job or if the new job has twice the value of the currently running job. It is easy to see that the competitive ratio of this algorithm is $O(\log_2 \mu_d) = O(\log_2(M/m))$.

6 Open Questions

The first question we leave open is simply to close the quadratic gap for our generic scheduling problem. The most desirable result would give a logarithmic lower bound. This would not only match the upper bound shown in Section 5 but would also match the other logarithmic upper bounds discussed in Section 1.1.

Although much of the recent work in online routing in communication networks addresses the limited capacity problem, most of the work on the more traditional load balancing problems relates to makespan minimization. Despite the general lack of attention, the limited capacity problem seems to be well suited to many of the applications for the variety of scheduling paradigms in the literature. This points out an important and largely unexplored area in online scheduling. For example, there has been little which addresses admission control problems related to either example (a) or (b) at the beginning of the introduction. Although the work of [10], [11] and [21] is an important step in this direction, their work addresses a very special cases. Thus, generalizing these results to apply to more complex models is an important and relevant new direction for research.

Acknowledgements

We are grateful to the hospitality of Zvi Galil and Columbia University that enabled us to carry out this work during the summer of 1994.

References

[1] J. Aspens, Y. Azar, A. Fiat, S. Plotkin, O. Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Symposium on the Theory of Computing*, pages 623–631, 1993.

[2] B. Awerbuch, Y. Azar, S. Plotkin. Throughput-competitive online routing. In *34th IEEE Symposium on Foundations of Computer Science*, 1993. 32–40.

[3] B. Awerbuch, Y. Bartal, A. Fiat, A. Rosen. Competitive non-preemptive call control. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 312–320, 1994.

[4] B. Awerbuch, R. Gawlick, T. Leighton, Y. Rabani. On-line admission control and circuit routing for high performance computing and communication. To appear in *Proc. 35th Annual Symposium on the Foundations of Computer Science*, 1994.

[5] Y. Azar, A.Z. Broder, A.R. Karlin. Online load balancing. In *Proc. 33rd Annual Symposium on the Foundations of Computer Science*, 1992, pages 218-225.

[6] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, O. Waarts. Online load balancing of temporary tasks. In *Workshop on Algorithms and Data Structures*, pages 119–130, 1993.

[7] Y. Azar, J. Naor, R. Rom. The competitiveness of on-line assignments. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 203–210, 1992.

[8] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, B. Schieber. Bandwidth Allocation with Preemption. Submitted to these proceedings.

[9] Y. Bartal, A. Fiat, H. Karloff, R. Vohra. New algorithms for an ancient scheduling problem. In *Proc. 24th ACM Symposium on Theory of Algorithms*, pages 51–58, 1992. To appear in *Journal of Computer and System Sciences.*

[10] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha. On-line scheduling in the presence of overload. In *Proc. 32nd Annual Symposium on the Foundations of Computer Science*, 1991, pages 100-110.

[11] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, F. Wang. On the competitiveness of on-line real-time task scheduling. In *The Journal of Real-Time Systems*, 4(1992)124-144.

[12] J.A. Garay, I.S. Gopal. Call preemption in communication networks. In *Proceedings of INFOCOM '92*, 1992.

[13] J. Garay, I.S. Gopal, S. Kutten, Y. Mansour, M. Yung. Efficient on-line call control algorithms. In *Proc. 2nd Israel Symposium on Theory of Computing and Systems*, pages 285 – 293, June 1993.

[14] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416-429,1969.

[15] D. R. Karger, S. J. Phillips, E. Torng. A better algorithm for an ancient scheduling problem. In *Proc. of the 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 132–140, 1994.

[16] G. Koren and D. Shasha. D^{over}: An optimal on-line scheduling algorithm for overloaded real-time systems. Technical Report 594, Courant Institute, New York University, 1992.

[17] G. Koren and D. Shasha. MOCA: A Multiprocessor On-Line Competitive Algorithm for Real-Time System Scheduling. *Theoretical Computer Science, Special Issue on Dependable Parallel Computing*, No. 128, pages 75-97, 1994.

[18] R. Motwani, S. Phillips, E. Torng. Non-clairvoyant scheduling. In *Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 422–431, 1993. Also to appear in *Theoretical Computer Science*, Special Issue on Dynamic and On-Line Algorithms.

[19] S. Phillips, J. Westbrook. On-line load balancing and network flow. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 402–411, 1993.

[20] K.K. Ramakrishnan, L. Vaitzblit, C. Gray, U. Vahalia, D Ting, P. Tzelnic, S. Glaser, W. Duso. Operating System Support for a Video-On-Demand File Service. Network and Operating System Support for Digital Audio and Video: Fourth International Workshop, Lancaster, UK. Springer Verlag, pages 225–236, 1993.

[21] G. Woeginger. On-line scheduling of jobs with fixed start and end times. In *Theoretical Computer Science* 130(1994)5-16.

[22] Special Issue on Asynchronous Transfer Mode. Int. Journal of Digital and Analog Cabled Systems, 1(4), 1988.