Parallel Scheduling Problems in Next Generation Wireless Networks

L. Becchetti[*], S. Diggavi [†], S. Leonardi[*], A. Marchetti-Spaccamela[*], S. Muthukrishnan[†], T. Nandagopal[‡], A. Vitaletti[*]

ABSTRACT

Next generation 3G/4G wireless data networks allow multiple codes (or channels) to be allocated to a single user, where each code can support multiple data rates. Providing fine-grained QoS to users in such networks poses the two dimensional challenge of assigning both *power (rate) and codes for every user. This gives rise to a new class of parallel scheduling problems. We abstract general downlink scheduling problems suitable for proposed next generation wireless data systems. This includes a communication-theoretic model for multirate wireless channels. In addition, while conventional focus has been on throughput maximization, we attempt to optimize the maximum response time of jobs, which is more suitable for stream of user requests. We present provable results on the algorithmic complexity of these scheduling problems. In particular, we are able to provide very simple, online algorithms for approximating the optimal maximum response time. This relies on resource augmented competitive analysis. We also perform an experimental study with realistic data of channel conditions and user requests to show that our algorithms are more accurate than our worst case analysis shows, and they provide fine-grained QoS to users effectively.*

1. INTRODUCTION

There is tremendous momentum in the wireless industry towards next generation (3G and beyond)[1] systems. These systems will not only migrate the existing voice traffic to a higher bandwidth platform, but are also expected to jump-start large scale data traffic. These emerging wireless systems[2] such as CDMA, wideband OFDM and multislot TDMA allow multiple codes (channels) to be allocated to users, in each of which traffic can flow in one of multiple rates. This provides them more flexibility than is available in current systems to manage and modulate the traffic. This also gives rise to novel parallel scheduling problems that we study in this paper. We use the terms code and channel interchangeably, since the terminology varies between CDMA, OFDM and TDMA systems. So, for instance, a code (channel) will denote a code when referring to CDMA, a frequency tone when referring to OFDM and a time slot for TDMA.

In a packet wireless cellular architecture each cell has a base station and is connected by a high-speed backbone to the Internet. Each base station handles all requests to and from mobile users within the cell, i.e., it handles both uplink (from mobile users) and downlink (to mobile users) requests.

Our focus in this paper is on the downlink channel performance, which is likely to be a major focus in emerging systems since data traffic is expected to dominate over time and data traffic typically tends to have asymmetrically large downlink demand. Existing wireline scheduling and resource allocation algorithms can not be directly applied to manage the downlink since wireless networks have unique characteristics, such as location dependent channel errors. Users in different regions of a cell experience different channel conditions resulting in location dependent dependent data and packet error rates. Unlike traditional scheduling scenarios, in a wireless environment, the scheduler must consider channel state in order to provide reasonable Quality of Service (QoS), and wireless systems have a variety of built-in capabilities to gather channel condition information for this purpose. In addition, random channel errors may result in poor performance of transport protocols such as TCP. Existing solutions to this problem propose intercepting the connection at the base station, creating two logical connections [2]. The base station thus acts as a proxy, and interprets the packets up to the transport layer in order to address random channel errors. In addition, in many commercial wireless networks proxy servers store-and-forward data to mobile users, and thus have information regarding the various requests in the system, including user request sizes. We will assume this context and address the parallel scheduling problems that arise in this setting.

1.1 Contributions

Our contributions are threefold.

- We abstract a general downlink scheduling problem in

[*]Universita' di Roma "La Sapienza", Rome, Italy. Email:{becchett,alberto,leon,vitale}@dis.uniroma1.it

[†]AT&T Shannon Labs, Florham Park, New Jersey, USA. Email:{suhas,muthu}@research.att.com

[‡]University of Illinois, Urbana-Champaign, Illinois, USA. Email: thyagu@crhc.uiuc.edu

[1]G: Generation. Current systems are 2G or 2.5G, where rudimentary data services are being deployed.

[2]CDMA: Code Division Multiple Access; OFDM: Orthogonal Frequency Division Multiplexing; TDMA: Time-Division Multiple Access. See [1] for more details.

next generation wireless data networks. This problem has many novelties. For example, we embody channel characteristics guided by communication theoretic considerations, and the properties of these channels get exploited in our scheduling algorithms. Second, we study QoS parameters related to per request behavior, in particular, we focus on optimizing response time per request. In contrast, prior work in wireless systems scheduling has typically focused on rate optimization (maximization) metrics (see for instance [3, 4, 5]).

- The scheduling problems that arise above are novel versions of multidimensional malleable task parallel scheduling. Here one can think of the number of channels as the number of processors and the malleability arises because the data rates depend on the allocation of number of channels and the power on each channel. We show these problems to be NP-complete and that they are also hard to approximate. However, we use resource-augmented competitive analysis, to derive simple, online algorithms that provably have good performance in approximating the optimal maximum response time of a job. We also present results for other QoS metrics, such as average and weighted response times.
- We present a detailed experimental study of our algorithms. Using real web server request logs and realistic 3G/4G system parameters, we show experimentally that our online algorithms are practical, and perform significantly better than our worst-case analyses indicate. We also examine which resource is most important to augment in a wireless setting.

1.2 Related Work

There has been a significant amount of work on scheduling problems over wireless channels. We study the downlink scheduling problem. The uplink scheduling problem is a complementary problem where the fundamental issues are quite different. See [6] and references therein for more details.

Typically resource allocation problems study per-user rate throughput. The rate optimization problem has been extensively studied for various wireless system with focus varying from maximizing overall throughput to providing a minimum throughput guarantee for all users. A good discussion of related work about throughput optimization and fairness in wireless data networks can found in [3, 4].

Job scheduling is very popular in the context of processor scheduling, and various algorithms have been proposed for different QoS metrics such as completion time, maximum response time and, weighted response time [7]. In the parallel scheduling literature, there has been significant work on scheduling of *malleable* tasks, see for example [8, 9, 10, 11, 12]. Our multidimensional malleable scheduling problem has not been studied previously.

In wireless networks, job scheduling has been addressed in the context of downlink *broadcast* scheduling [13]. There a single transmission may satisfy multiple users or requests which is a model that is applicable in many special purpose systems. In contrast, we focus on unicast scheduling where each transmission is destined to a unique request which applies to general purpose wireless data networking. In a recent work, downlink unicast scheduling in CDMA systems was studied [14]; this is close to our work in spirit. However, they assume a linear rate model for the physical layer which is not accurate. Also, they do not have any upper bound on the number of available codes; hence, they study the problem of allocating power only. We have studied the nuances of allocating both power and codes, which is more suitable. Finally, we have provided a thorough competitive analysis of the online algorithms, in particular, using the resource augmented analysis; this is the first provable result known for any of the online scheduling problems, including the ones in [14].

Roadmap:. We present the communication channel model, and abstract our scheduling problem in Section 2. In Section 3, we present a theoretical study showing the structure and the complexity of these problems. In Section 4, we present our main algorithmic results, namely, simple online algorithms and present our augmented-resource based analyses. In Section 5, we present our experimental results. We conclude the paper in Section 6. We have obtained a number of algorithmic results for optimizing other per request QoS metrics such as (weighted) average response times which we have included in Section 4.2. Due to space constraints, many of the proofs are given in [1].

2. PROBLEM FORMULATION

In this section, we will describe effects of the wireless communication medium on the transmission rates for each user, and also formulate parallel scheduling problems that arise in next generation wireless networks. (See [1] where we describe some details about next generation wireless proposals which will fit our problem abstraction.)

2.1 Communication Channel Model

In wireless systems, channels have variable attenuation depending on the geographic location of the users. For example, in the CDMA case, all codes assigned to a user present the same attenuation, which depends on that user's location. This is mainly due to multipath impairments and radio propagation losses. Say the base station (BS) is communicating with n mobile users. The physical channel (code) attenuations of the users are denoted by $\bar{g}_1, \bar{g}_2, \cdots, \bar{g}_n$ respectively; each $\bar{g}_i$ is a scalar parameter called the *physical* gain. If the BS transmits power p_i to a user i, the signal-to-interference-plus-noise ratio (SINR) is given by $SINR = \frac{\bar{g}_i p_i}{\sigma^2}$, where σ^2 is the total noise power (including interference) [15]. SINR determines the rate of transmission of packets to the user. In particular, the rate $r_{bps}(\cdot)$ as a function of the SINR is a concave logarithmic function [16],

$$r_{bps}(SINR) = \bar{W} \log_2(1 + \frac{x}{\Gamma}) \quad (1)$$

where $r_{bps}(SINR)$ is the rate in bits per second, $\bar{W}$ is the spectral bandwidth used, and Γ is dependent on the coding gain from the physical layer error-correcting code [16]. Both Γ and $\bar{W}$ are system parameters, which for our purposes will be constants. Therefore, for a particular user i, the number of bits received over a period of time τ and over a single channel, obtained as a function of the power p_i allocated to the user on a single channel (code), is given by

$$r_i(p_i) = \bar{W}\tau \log_2(1 + \frac{\bar{g}_i p_i}{\sigma^2 \Gamma}) \quad (2)$$

The rate vs. SINR curves for next-generation wireless systems closely approximate the convex function described by this equation (see for example [17]). This rate function already embodies the effect of variable rate error-correcting coding schemes in the physical layer, as is typical in next generation wireless systems [16, 17, 18]. Therefore, we will use this equation for rate calculations in our scheduling problems. For notational convenience we will denote $g_i = \frac{\bar{g}_i}{\Gamma\sigma^2}$ as the *channel gain* and we will set $W = \tau\bar{W}$ yielding $r_i(p_i) = W\log_2(1 + p_i g_i)$. Observe that $SINR$ is implicitly contained in the term $p_i g_i$.

2.2 Abstract Scheduling Problem

The base station has a total power P to transmit. Time is assumed to be partitioned into equal width windows called *time slots*[3] (or frames), whose width is τ. We also assume that there are a total of C codes which can be assigned to users in a time slot. If user u makes a request i, then we say that the gain of request i is the same as the gain of the user u, $g_i = g_u$. Requests[4] arrive in the system over time at the beginning of time slots; requests/jobs are for non-real time traffic such as browsing, downloads, etc., i.e., file transfers. The size s_i (in bits) and the channel gain g_i of the user who made the ith request are known when the request arrives at time a_i. The arrival time is also known as *release time.* We will assume that the channel conditions of the users are constant over the scheduling period. Although this is a simplification, it holds in realistic cases[5]. The scheduling problem is to determine an assignment of power and codes to each user in each time slot, for the required QoS criterion. Formally, we have online job arrivals which specifies job size, and users time of arrival of each job, with the scheduling algorithm producing the following output.

Output: The set $\mathcal{C}_u(t)$, the set of codes assigned to user u at time t, and $p_u^{(i)}(t)$, the power assigned to user u at time t to each code $i \in \mathcal{C}_u(t)$. The assignment must satisfy the following conditions.

- *Total Code and Power Constraints.* Obviously, for any time slot t, $\sum_u |\mathcal{C}_u(t)| \le C$ and $\sum_u \sum_{i\in\mathcal{C}_u(t)} p_u^{(i)}(t) \le P$.
- *Discrete Rate Set* : Only a discrete set of rates (equivalently, minimum power per discrete rate) is allowed. These rates denoted $R(1), R(2), \ldots$ have the property[6] that $\frac{R(i)}{R(i-1)} \le 2$.
- *Request Completion*: All requests get the requested data size, that is, if s_u and $R_u(t)$ denote the size of request u and $R_u(t)$ the rate in time slot t, then we need

$$s_u = \sum_t R_u(t) = \sum_t \sum_{i\in\mathcal{C}_u(t)} r_i(p_i) \qquad (3)$$

where $\mathcal{C}_u(t)$ is the set of codes assigned to user u in time slot t, and $r_i(p_i)$ is calculated using (2), subject to the discrete rate set constraint above.

Our goal is to *minimize the maximum response time* or *max-flow* [19], where response time for request i is $c_i - a_i$, if request i is completed by time c_i.[7] We will focus on this goal for most of the paper. We do prove results for certain related metrics such as minimizing total weighted response time, but those results appear in Section 4.2 with proofs in [1].

We assume requests may be served over several time slots with different sets of codes at each time slot. In standard scheduling terminology [19], this corresponds to requests being *preempted* (*i.e.*, stop processing a request, process other requests, and resume the original request) and *migrated* (*i.e.*, assign sets of codes to a user that differ from one time slot to another)[8].

There are two basic variants of our problems, namely *offline* or *online.* In the offline version, all request arrivals are known ahead of time. The offline case is of theoretical interest and is mainly useful to quantify the benefit to be accrued from scheduling. In the online case, requests arrive over time and the scheduling algorithms have to take their decisions without knowledge of future requests. The performance of the online algorithms is measured in comparison to the offline case as in standard competitive analysis.

A *malleable resource* is one in which giving more of it (say, more processors) improves performance in the other (runs faster). Therefore, our problem is a version of a multidimensional malleable resource problem. Prior work involves malleable scheduling of parallelizable jobs [8, 10] or scheduling a mixture of malleable and non-malleable jobs [20, 9], but none address our problem. For example, in our case, the processing time of a job depends on two variables, the number of codes and the assigned power per slot; both these resources are malleable, and they have a strongly (nonlinearly) coupled effect on data rate through Equations (2)-(3). We are not aware of prior work on scheduling multiple malleable resources with preemption and migration as we do, that minimizes the maximum response time under the coupling constraints of Equations (2)-(3). Therefore, our work is different from [21] since our focus is on maximum response time, with preemption and online job arrivals. Finally, to the best of our knowledge, the resource augmented competitive analysis for scheduling with multiple malleable resources has not been previously presented.

3. UNDERSTANDING THE SCHEDULING PROBLEMS

3.1 Some Structural Observations

Here, we state and prove some properties of the communication channel. They will be invoked later in proving our

[3]We will use time and time slot interchangeably when no confusion arises.

[4]The terms: requests, jobs and users, will be used interchangeably.

[5]If time scale of the scheduler is several seconds, and if the users do not have very high mobility, then the channel conditions will be static over this time scale [15].

[6]This relationship holds for existing next generation wireless data system proposals like cdma2000 and HDR, which have rate set of {38.4, 76.8, 102.6, 153.6, 204.8, 307.2, 614.4, 921.6, 1228.8, 1843.2, 2457.6} kilobits per second (kbps). The factor 2 is not sacrosanct. If the discrete rates are more spread out, but bounded by some constant, all our results will apply with minor changes in the claimed bounds.

[7]This is also sometimes called flow time in literature.

[8]A more detailed model may distinguish some codes to be more preferable than the others from one time slot to another to insure intercell interference avoidance, an issue we do not consider in this paper.

main results.
Continuous Power (Rate) Case: First, we consider case where the rates are not discrete and are a monotonic function of power, as in Equation (2). Concavity of the rate with respect to p in (2) implies that if we assign $c \geq 1$ codes to a user u, then *it is optimal to divide the total power p allocated to that user equally among the codes assigned* as summarized below.

LEMMA 3.1. ***(Equipartition of power)*** *Given c codes and power p to a user, the rate r is maximized for* $p_i = p/c,\ i = 1, \ldots, c$.

Using Lemma 3.1 we can write the rate obtained by a user given c codes and p total power as,

$$R(p, c) = Wc\log(1 + \frac{gp}{c}). \quad (4)$$

Discrete Power (Rate) Case: When we have a discrete rate set, the actual rate obtained on each code is given by the highest discrete rate[9] which is below $W\log(1 + \frac{gp}{c})$. Therefore, the difference between the continuous rate and the discrete rate case, depends on the discrete rate set available. Hence,

FACT 3.1. *If* $\frac{R(j)}{R(j-1)} \leq 2$ *for the discrete rate set* $\{R(j)\}$, *then for any continuous power allocation p there exists a discrete rate* $R(l)$ *such that* $R(l) \geq \frac{1}{2}r_i(p)$, *where* $r_i(p)$ *is given by (2).*

3.2 Computational Hardness

In order to understand the challenge of the problem further, let us consider the offline complexity of the scheduling problems. If power (rate) values are required to be drawn from a discrete set, the problem in its simplest instance is the bin packing problem and hence it is NP-complete [22]. We focus on the more challenging case when code is discrete, as usual, but the power (and hence rate) is allowed to take any continuous value. In order to prove the hardness of this problem, we will consider the version of the problem in which ith request has arrival (release) time a_i, deadline d_i and size s_i in bytes. Using this, we can state the following theorem whose proof is given in the [1].

THEOREM 3.1. *If the number of codes assigned to each user is integral, then it is NP-complete to compute a feasible schedule for the problem of meeting deadlines even if all users have the same channel gain, a common release time, a common deadline and the power assigned to each code is not restricted to a discrete set of values.*

The result above in fact shows the problem to be NP-complete in the strong sense (see [22] for definition and significance). Although we have shown this hardness result only for the deadlines problem, it is easy to see that this immediately gives the hardness of other the scheduling problem we have, namely, minimizing the maximum response times. Finally, notice that the result holds independently of how rates are affected by the use of multiple codes, since the hardness is proved even for the restrictive case when each request gets only one code over all time slots.

We can also show that this problem is hard to approximate. This is summarized in the following theorem whose proof is given in [1]. Therefore, in order to prove approximation results we require to use the techniques of resource augmentation as done in Section 4.

THEOREM 3.2. *For every* $c > 0$, *it is NP-hard to compute a c-approximation of the maximum response time.*

3.3 Offline Scheduling Problem

We study the offline version *i.e.*, when all arrival times are known apriori. Using this, we will get lower bounds on optimum values of certain QoS metrics which will be a benchmark to compare against online algorithms.
Deadlines Scheduling Problem: Here, each request j has an arrival time a_j as well as a deadline d_j. As before, for each request j, at time a_j we know its size s_j and the channel gain g_j. The goal is to merely test feasibility, *i.e.,* determine if there is a valid schedule that meets all deadlines. This problem is the technical core of many other scheduling problems.

We can write the solution to the deadlines scheduling problem as a combinatorial optimization program as shown in Table 1. Here, $c(j, t)$ denotes the number of codes assigned to user j in time slot t and let $p(j, t)$ be the total power assigned to user j in time slot t over all the codes. This is called as the *time indexed program* in the table.

Since $c()$ and $p()$ take on only discrete values, even to check feasibility is an NP-complete problem as proved earlier. Hence, we relax the variables to be continuous by allowing $c(j, t)$ and $p(j, t)$ to be fractional, resulting in the *Fractional Time Indexed Program.* The following is proved in [1] and is based on showing that the constraint set in the interval indexed program is convex.

THEOREM 3.3. *There exists a pseudo-polynomial time algorithm to solve the Fractional Time Indexed Program.*

Interval Indexed Program: The relaxation of the integer programming problem to the pseudo-polynomial time algorithm, still results in a problem size of $O(nT)$ variables, where n is the number of requests and T is the total length of the schedule. Next we consider decreasing the number of variables used in the convex program. We will define a new program below called the *interval indexed program.*

An *event* is either the arrival or the deadline of a request in the system. Consider the sorted list of the events $t_1, \ldots, t_K$. We divide the time scale into *intervals* where an interval is the time period between any two consecutive events, that is interval I_k contains $[t_k, t_{k+1})$. For n requests, the total number of intervals is at most $2n$. We will look for *sliver* solutions, that is, ones in which for each interval I, each user j gets power $p(j, t)$ and $c(j, t)$ for $t \in I$ that remains constant for all $t \in I$, that is, $p(j, t_1) = p(j, t_2)$ for $t_1, t_2 \in I$ and likewise for $c()$. Let $c(j, k)$ be the fractional number of codes and $p(j, k)$ be the fractional power assigned to job j in interval k *per time slot.* Let a_j^{-1} denote the interval at the beginning of which job j arrives in the system, and d_j^{-1} be the interval at the end of which its deadline lies.

The fractional programming formulation for solving the scheduling problem with slivers is given in the Table 1.

[9]Note that we make a regularity assumption that the user gains are such that the lowest discrete rate $R(1) \leq W\log(1+gP)$, *i.e.* by allocating all the resources to the user there exists a feasible discrete rate. In practice, error-correcting codes can be used over a group of codes to increase the dynamic range of user gains that fall into the feasible region.

Theorem 3.4. *The time indexed program has a feasible solution if and only if the interval indexed program has a feasible solution. It can be solved in time polynomial in n, C using convex programming.*

The proof is given in [1]. The result above exposes an interesting structural property of the interval indexed convex program, *i.e.*, the structure that sliver assignment of power and code to requests is optimal in the fractional case.

Using the Deadlines Scheduling Problem: Given that the feasibility of the (relaxed) time-indexed program can be solved efficiently, we can use it to solve the offline maximum response time problem nearly optimally. We do this by guessing a target response time F, and checking the feasibility of a deadline scheduling problem with deadlines $a_i + F$. By doing a binary search on the target response time value, we get an efficient (polynomial time) algorithm to optimize the maximum response time. Indeed the same approach works for optimizing quality of service criteria such as $\max_i f(c_i - a_i)$ for any monotonic increasing function f.

4. ONLINE HEURISTICS

In this section we present our main algorithmic results, namely, a set of online algorithms for optimizing the metrics of our interest. As is standard, we measure the performance of an online algorithm using the ratio of the value of the objective function (here, *max-flow*) achieved by the online algorithm and the optimal value, which can be computed offline.

In our analysis we also use *resource augmentation* [23]. That is, we compare the optimum (i.e. the solution found by adversary) with the value of the solution found by the online algorithm when it is provided with more resources than an adversary who can serve it optimally.

Formally, we say that algorithm A for our scheduling problem is an $(\alpha, \beta, \gamma, \delta)$ approximation if it provides a β approximation of the optimum when the sizes of user requests are scaled down by a factor α and the number of codes (the power) used by the algorithm is at most γ (δ, respectively) times the number of codes (the power, respectively) used by the optimum solution to serve the original input sequence.

4.1 Minimizing the Maximum Response Time

We first develop our analysis for the "continuous" rate case (Theorem 4.2) and then analyze the case when only discrete rates per code is allowed (Theorem 4.4). It is well known in processor scheduling literature [19] that the online algorithm Earliest Release Time (ERT) or First In First Out (FIFO) is optimal for minimizing the maximum response time on a single machine and is a 3 approximation algorithm on parallel machines. Therefore, it is natural to ask how it would perform in our case. The simple FIFO strategy in our case allocates all the codes and power to one user at a time till the user completes the job. Using the "Equipartition of power" lemma (Lemma 3.1), this translates into giving the user a power per code of P/C in consecutive time slots that the user completely occupies. Therefore, we study the online scheme where each user is given a power P/C per code and is served by a FIFO scheduling discipline. We call this scheduling discipline FIFO($\frac{P}{C}$). Given an instance $\mathcal{I}$ of continuous job arrivals, we denote the maximum response time of the scheduling discipline FIFO($\frac{P}{C}$) by $f^{FIFO}(\mathcal{I})$ and the optimal discipline has a maximum response time of $f^{OPT}(\mathcal{I})$. We first we show a negative result (with proof in [1]), which demonstrates that such a strategy could be arbitrarily worse than the optimal strategy.

Theorem 4.1. *For any $M > 0$ there is an instance $\mathcal{I}$ of continuous job arrivals, such that $f^{FIFO}(\mathcal{I})/f^{OPT}(\mathcal{I}) > M$.*

In spite of the negative results above, we can show that FIFO($\frac{P}{C}$) is able to achieve the optimum if every request is reduced to 50% of its original size. In the sequel, given an instance $\mathcal{I}$ of the problem, we denote by $\mathcal{I}'$ the instance in which each job size s_j in $\mathcal{I}$ has been reduced to $s_j/2$. We also denote by $p_j^{OPT}(\mathcal{I})$ and $k_j^{OPT}(\mathcal{I})$ the overall power and codes assigned to the jth job of instance $\mathcal{I}$ by the optimum. Denote by $k_j(\mathcal{I}')$ the overall code assignment to job j by the scheduling discipline FIFO($\frac{P}{C}$) applied to $\mathcal{I}'$. In order to prove our result we need the following lemma whose proof is given in Appendix A.

Lemma 4.1.

$$\frac{k_j(\mathcal{I}')}{C} \leq \max\left\{\frac{p_j^{OPT}(\mathcal{I})}{P}, \frac{k_j^{OPT}(\mathcal{I})}{C}\right\} \tag{5}$$

The algorithm schedules the jobs in order of release time, assigning a number of codes k_j^{red} with power allocation P/C until 50% of the original demand is met. Its operation can be summarized as follows:

1. When a request j is presented, find the number $k_j(\mathcal{I}')$ of codes needed to complete demand s_j^r reduced by 50% with $\frac{P}{C}$ power per code.
2. When a job is completed select the pending user request, if any, with earliest release time a_j.

Observe that a job may be served using more than C codes, over more time slots.

Theorem 4.2.

$$f^{FIFO}(\mathcal{I}') \leq f^{OPT}(\mathcal{I}) + 2, \ \forall \mathcal{I}. \tag{6}$$

Proof. Consider the request r achieving the maximum flow time for the scheduling discipline FIFO($\frac{P}{C}$) applied to instance $\mathcal{I}'$. W.l.o.g., we assume that request r is the last request presented to the algorithm. Request r has been released in slot t_r and completed in slot $C_r(\mathcal{I}')$ in the algorithm's solution. Denote by t (the renewal time) the last slot in which all requests that have been presented before time t have been completed by slot t. We can restrict our attention to the subset of user requests, denoted by $\mathcal{J}' = \{j \in \mathcal{J} | a_j \geq t\}$, that have been presented at or after slot t, since these are the only requests that contribute to the flow time of request r. The completion time for request r using the FIFO($\frac{P}{C}$) discipline on instance $\mathcal{I}'$ is at most $C_r(\mathcal{I}') \leq t + \left\lceil \frac{\sum_j k_j(\mathcal{I}')}{C} \right\rceil$, where $k_j(\mathcal{I}')$ is the number of codes required to complete job j on instance $\mathcal{I}'$. Denote by s the user request completed last in the solution of the optimum on instance $\mathcal{I}$. Request s has been released at some time $t_s \leq t_r$ and therefore, $C_s^{OPT}(\mathcal{I}) - t_s \geq t - t_s + \Psi$, where

$$\Psi = \max\left\{\left\lceil \frac{\sum_j p_j^{OPT}(\mathcal{I})}{P} \right\rceil, \left\lceil \frac{\sum_j k_j^{OPT}(\mathcal{I})}{C} \right\rceil\right\}.$$

Now, we have

$$\begin{aligned} f^{FIFO}(\mathcal{I}') &= C_r(\mathcal{I}') - t_r \qquad (7)\\ &\leq \; t - t_r + \left\lceil \frac{\sum_j k_j(\mathcal{I}')}{C} \right\rceil \overset{(a)}{\leq} t - t_s + \Psi \\ &\leq \; C_s^{OPT}(\mathcal{I}) - t_s + 2 \leq f^{OPT}(\mathcal{I}) + 2, \end{aligned}$$

where (a) follows from Lemma 4.1 giving us the result.

□

This result shows that by reducing the demand, we can prove a positive result on FIFO($\frac{P}{C}$). As seen in the following theorem (whose proof is in [1]), if we allow for resource augmentation, a positive result can be shown on FIFO($\frac{P'}{C'}$).

THEOREM 4.3. *Denote by $f^{FIFO_{Aug}}(\mathcal{I})$ the maximum response time of the scheduling discipline* FIFO$_{Aug}(\frac{P'}{C'})$ *applied to $\mathcal{I}$ where the power and number of codes per time slot have been augmented to P' and C' respectively. Then, $\exists\, P' \leq 2P$, $C' \leq 2C$ such that,*

$$f^{FIFO_{Aug}}(\mathcal{I}) \leq f^{OPT}(\mathcal{I}), \ \forall \mathcal{I}. \qquad (8)$$

We now show how to transform our algorithms for the continuous case into algorithms for the discrete case. Recall that in the continuous case we assign power P/C to every code, but this may correspond to a non-feasible transmission rate at the receiver for some specific user. To move from the continuous to the discrete case, we need to round the power assignment to a value that sustains one of the discrete transmission rates[10].

We will implement a rounding scheme that allows us to turn a solution for the continuous case into a solution for the discrete case. We perform two different kinds of rounding of a power z assigned to a code:

1. *Round up*: If there exists a power $\overline{z}_1 \in (z, 2z]$ corresponding to a discrete rate, then assign power $\overline{z}_1$ to the code.

2. *Round down*: If there exists a power $\overline{z}_2 \leq z$ corresponding to a discrete rate, then assign power $\overline{z}_2$ per code.

Then for each user we choose the rounding that gives the higher rate, if the user were given all the resources *i.e.,* all the power and codes. We denote by FIFO$_{disc}(\frac{P}{C})$, the scheduling discipline obtained by taking the discipline FIFO($\frac{P}{C}$) and applying the above rounding procedure. We also define by $f^{FIFO_{disc}}(\mathcal{I})$, the maximum response time of FIFO$_{disc}(\frac{P}{C})$ on a online job arrival instance $\mathcal{I}$. It is easy to see that the following result holds (its proof is given in [1]).

LEMMA 4.2. *The allocation scheme for the discrete case satisfies all users demands.*

[10]For the P/C allocation we impose a further regularity condition that the lowest discrete rate $R(1) \leq W \log(1 + \frac{gP}{C})$, *i.e.* there is a feasible discrete rate below this power allocation. Though this is perhaps a little more stringent than required, it makes the analysis simpler. As before this restriction can be removed in practice by using error-correcting codes on a group of codes, so that the combined rate is a feasible discrete rate.

Now, we show that the approximations shown for the continuous case can be translated to the discrete rate by additional resource augmentation. The proof idea is to show that additional power and codes needed, for the rounding procedure above to be as good as the continuous case, is bounded.

THEOREM 4.4. *Let* FIFO$_{disc}(\frac{P'}{C'})$ *be applied to $\mathcal{I}'$, where the power and number of codes per time slot have been augmented to P' and C' respectively. Then, $\exists\, P' \leq 2P$, $C' \leq 2C$ such that,*

$$f^{FIFO_{disc}}(\mathcal{I}') \leq f^{OPT}(\mathcal{I}) + 2, \ \forall \mathcal{I}. \qquad (9)$$

PROOF. Let $\mathcal{K}_1$ ($\mathcal{K}_2$) denote the number of codes whose associated power was rounded up (respectively down) in a particular time slot l. Clearly, $|\mathcal{K}_1| + |\mathcal{K}_2| \leq C$. Let the power allocated on each code i after rounding be denoted by $p^{rnd}(i)$, and clearly $p^{rnd}(i) \leq 2P/C, i \in \mathcal{K}_1$ and $p^{rnd}(i) \leq P/C, i \in \mathcal{K}_2$. Now, suppose in each time-slot, for every code $i \in \mathcal{K}_2$ we assign two codes with power $p^{rnd}(i)$, and for each code $i \in \mathcal{K}_1$ (whose power was rounded up) we assign one code. Clearly such an allocation will meet the same demand as the continuous rate FIFO($\frac{P}{C}$) schedule. Hence the schedule is equivalent to the the continuous rate FIFO($\frac{P}{C}$) schedule. Therefore using this and Theorem 4.2 the result (9) can be obtained. The only question that remains is how much resource augmentation was done to obtain this. In the new allocation we have used $P' = \sum_{i \in \mathcal{K}_1} p^{rnd}(i) + 2 \sum_{i \in \mathcal{K}_2} p^{rnd}(i)$ total power and $C' = |\mathcal{K}_1| + 2|\mathcal{K}_2|$ total codes. But we have,

$$\begin{aligned} P' &\leq 2\frac{P}{C}|\mathcal{K}_1| + \frac{P}{C}2|\mathcal{K}_2| \leq 2P \qquad (10)\\ C' &= |\mathcal{K}_1| + 2|\mathcal{K}_2| \leq 2C. \end{aligned}$$

Hence the new allocation uses at most a power $P' \leq 2P$ and a number of codes $C' \leq 2C$. □

We can also extend the result in Theorem 4.3 to the discrete rate with more resource augmentation, giving us the following result.

THEOREM 4.5. *If the scheduling discipline* FIFO$_{disc}(\frac{P'}{C'})$ *is applied to $\mathcal{I}$ where the power and number of codes per time slot have been augmented to P' and C' respectively. Then, $\exists\, P' \leq 4P$, $C' \leq 4C$ such that,*

$$f^{FIFO_{disc}}(\mathcal{I}) \leq f^{OPT}(\mathcal{I}), \ \forall \mathcal{I}. \qquad (11)$$

4.2 Other QoS crieria

Although we focused our attention on minimizing the maximum response time in this paper, we can extend our results to other optimization criteria such as *minimize total weighted response time*, $\sum_i w_i(c_i - a_i)$, where arbitrary weights w_i are specified for each request i. In particular we consider arbitrary weights w_i and for average flow time (where $w_i = \frac{1}{n}$, for n jobs). For average flow time, we analyze the Shortest Remaining Processing Time (SRPT) which is an optimal algorithm for the scheduling problem of minimizing the average flow time on a single machine [24]. For weighted response time, we analyze Highest Density first (HDF), which at any time t schedules the pending request with maximum ratio w_j/s_j, which is defined as the *density.* The following results are proved in [1].

THEOREM 4.6. *Algorithm SRPT achieves the optimum average flow time if every user demand is reduced to $\frac{1}{2(1+\epsilon)}$ of the original demand.*

THEOREM 4.7. *For any $\epsilon > 0$, Highest Density first is an $\frac{1+\epsilon}{\epsilon}$ approximation for minimizing the weighted flow time if every request is guaranteed for a fraction $\frac{1}{2(1+\epsilon)}$ of the original demand.*

5. SIMULATION STUDY

In this section, we study the performance of our online and offline algorithms experimentally.

5.1 Online Algorithms

The $FIFO(\frac{P}{C})$ algorithm was described in Section 4. We call this algorithm *FIFO-continuous*. Essentially, this algorithm allots P/C power to each code, and job requests are then scheduled in the order of their arrival.

The rounding procedure for converting the continuous power (rate) algorithm to a discrete power (rate) algorithm was described in Section 4.1. Since rounding the rates might result in some power and/or codes to be unused in a slot, we design discrete-rate online algorithms to minimize this potential waste of resources in order to reduce the maximum response time. We have developed three online discrete-rate algorithms, which we call *FIFO*, *2D-FIFO*, and *2D-PIKI*. Given a job, the power per code corresponding to the discrete bit rate is the same for all of these algorithms. They differ only in the way the jobs are selected for receiving service.

FIFO: This is the traditional FIFO algorithm. The request i currently in the system that has the earliest release time a_i is always selected.

2D-FIFO: The request i currently in the system that has the earliest release time a_i has higher priority over other job requests. However, if this job i leaves power/codes unused in that time-slot, other jobs j in the system are considered in the non-decreasing order of their release times a_j.

2D-PIKI: The request j currently in the system that has the highest value of power per code p_j is selected. If this job leaves power/codes unused in that time-slot, other jobs j in the system are considered in the non-increasing order of the power per code p_j. This scheme aims to achieve a better packing in each time slot, in order to reduce the completion time.

Due to discrete nature of the rate set, in certain slots the scheduler may have some codes k^{extra} and some power p^{extra} that cannot be assigned to any job in the system, since the power per code $p_j > p^{extra}$ for all jobs j. In such a situation, the scheduler will choose the first job that received service in the slot and give it the best possible discrete rate with the remaining power and codes. This is applicable to all the three algorithms described above.

Note that no algorithm guarantees that all the power and codes will be used in every slot. Therefore, we expect to see differences between the *FIFO-continuous* algorithm and the three discrete-rate algorithms. In the remainder of this section, we will quantify the differences through simulations.

5.2 Channel Specifications

We adopt the channel specifications similar to 3G system proposals [17, 18] for our channel model.[11]

We perform experiments on a single cell and abstract the effect of out-of-cell interferers into a decrease in SINR values. The peak power available at the base station was chosen to be $P = 40W$, while the maximum number of channels was chosen as $C = 16$. The power attenuation factor $\bar{g}_u$ for user u is modeled with two components: (a) shadow loss component S, which is a log-normal shadowing variable, and (b) path loss components $P = 1/d^{\alpha}$, where d is the distance between the base station and the user and α is the distance loss exponent. We chose $\alpha = 3$, giving $\bar{g}_u \propto S/d_u^3$.

The parameters to be used for the rate calculation given in Equation (2) were chosen as follows: $\tau = 1.67$ milliseconds, $\bar{W} = 76.8KHz$ and $\Gamma = 4.7dB$. We operated over an SINR range from -15dB/Hz to 15dB/Hz. The discrete rate set used is a set of 15 rates : {2.4, 4.8, 9.6, 19.2, 38.4, 76.8, 102.6, 153.6, 204.8, 307.2, 614.4, 921.6, 1228.8, 1843.2, 2457.6} Kbps. Under these restrictions, the maximum data rate for a mobile user in the cell will range from 10 Kbps to 2 Mbps.

5.3 Data Sets and Simulation Tools

The traces used in the experiments are derived from a single web-proxy server. For comparing the online algorithms with the offline optimum, we used traces that consist of up to 100 jobs that arrive over a period of 1 minute[12]. To evaluate the performance of various online algorithms under heavy demand, we use traces consisting of 4000 jobs arriving over a period of 35-40 minutes, where the requests are generated by 100 users in the cell. In all of the traces used, the minimum request size was 40 bytes, the maximum request size was 500 kilobytes, with mean request sizes ranging from 20 - 34 kilobytes. The average inter-arrival time of requests in the traces is 500 - 600 milliseconds.

In order to compute the optimum flow in the offline case, we used an optimization tool called $LOQO$ [25]. Our online algorithms were evaluated using a custom-built simulator.

5.4 Experiments

We performed two kinds of experiments to evaluate our algorithms. The first set of experiments validate our theoretical results and demonstrate some interesting properties of the online algorithms, while the second experiment was designed to measure the average-case performance of our algorithms.

5.4.1 *Online Heuristics*

In this section, we evaluate the different online algorithms and compare their performance against the offline optimal algorithm. We used small web traces, with 100 jobs arriving over a period of 1 minute, for computing the convex programming lower bound for max-flow (see Section 3.3), which we denote by OPT. The job requests are for users who are distributed uniformly in the cell. We present the max-flow results for four such traces along with the results

[11]We would like to emphasize that our algorithms are applicable to all systems that support multiple channels and multiple rates. Such systems include the various next-generation wireless data networks.

[12]The small size of these traces was primarily due to the computational restrictions on finding the optimal flow.

for the online heuristics in Table 2: all max-flow values are in terms of slots.

It can be seen that the online algorithms perform very close to the optimal, on the average. From the table, we also see that *2D-FIFO* performs the best among the three discrete-rate algorithms. In addition, the discrete algorithms always appear to perform worse than the continuous version.

These inferences continue to hold true in most instances, as we will show in the subsequent examples. We simulated the three algorithms *FIFO-continuous*, *2D-FIFO* and *2D-PIKI* on 36 traces of 4000 requests each generated by 100 users distributed over the cell. Figure 1 shows the values of the max-flow computed for all the 36 traces by the three algorithms. One can still observe that *2D-FIFO* is very close to *FIFO-continuous* on all traces, while *2D-PIKI* performs the worst.

Though these results hold on the average, we give a cautionary note that there can be examples where this average behavior is violated. We designed synthetic traces where *2D-PIKI* performs as well as the optimal algorithm and better than the *2D-FIFO* algorithm.

5.4.2 Resource Augmentation

In this set of experiments, we will examine the amount of resource augmentation needed for a discrete-rate algorithm to achieve the same max-flow as *FIFO-continuous* and compare it to theoretical bounds given in Theorems 4.4 and 4.5.

Using the same set of 36 web traces described in Section 5.4.1, the scheduling algorithms were provided with augmented power in steps $\{P, 1.25P, 1.5P, 1.75P, 2P\}$ and augmented number of codes in steps $\{C, 1.5C, 2C, 3C\}$.

In the first experiment we measure the worst case ratio between the max flow-time of *2D-FIFO* and *FIFO-continuous*. We tested the *2D-FIFO* algorithm, since it outperforms the other algorithms in the average case. The results of the first experiment are summarized in the 3-D graphic of Figure 2. We observe on all traces that *2D-FIFO* obtains a max-flow time lower than *FIFO-continuous* with power augmentation factor 1.5. We also found that a code augmentation factor of 4 is needed (without power augmentation) to obtain max-flow time lower than *FIFO-continuous*. Therefore augmenting codes is not as efficient as augmenting power. Another way to observe this is by examining (4) where the rate is $R(p,c) = Wc\log(1+\frac{gp}{c})$. Here one can see that $R(p,c) \leq Wgp, \forall c$, hence even with a large number of codes, the rate is bounded above. In contrast, the rate is an unbounded function of p.

In a second experiment, for each combination of augmented power and codes we reduce the demand in steps $\{s_i, 0.95s_i, 0.9s_i, 0.85s_i, 0.8s_i, 0.75s_i, 0.7s_i, 0.6s_i\}$ until we find the Max-flow for the reduced demand in the discrete case to be lesser or equal than the Max-flow computed by *FIFO-continuous* with power P and codes C at 100% of the demand. The lower hull over all traces of the reduced demand for each combination of augmented code and power was taken. This represents the maximum demand reduction for a given combination of augmented resources, as shown in Figure 3.

Two observations can be made here:

(a) *The average case is better than the worst-case.* In Figure 3 we observe that if power is augmented 1.25 P , then the Max-flow in the discrete case with 95% of the original demand equals that of the continuous case with power P.

(b) *Code augmentation and power augmentation are not the same.* As can be seen from Figures 2 and 3, the asymmetric nature of the graphs tells us that over-provisioning codes is not very efficient compared to over-provisioning of power.

In conclusion, our experimental results demonstrate that the discrete-rate algorithms proposed in this section perform closer to the continuous-rate case than the worst case analyses indicate. We also show that resource augmentation, in particular, power augmentation, will considerably enhance the performance of discrete algorithms.

6. DISCUSSION

We have formulated and studied new parallel scheduling problems in emerging 3G/4G wireless systems. We proposed simple, online algorithms that utilize the multicode, multirate feature of 3G/4G wireless networks for scheduling user requests. Our analyses involves resource-augmentation. Our experimental results show that in practice, the proposed algorithms perform much better than our worst case analyses predict. Therefore, we expect our results to find uses in providing fine-grained QoS to the users in next generation wireless data networks. This work should serve to initiate study of multidimensional malleable parallel scheduling problems under different metrics, some of which may be relevant in next generation wireless systems. We are also not aware of other experimental studies of scheduling algorithms that explore resource-augmentation needed in the analysis as done in this paper.

7. REFERENCES

[1] L. Becchetti, S. Diggavi, S. Leonardi, A. Marchetti-Spaccamela, S. Muthukrishnan, T. Nandagopal, and A. Vitaletti, "Parallel scheduling problems in next generation wireless networks," DIMACS Technical Report # 2002-29, available from http://dimacs.rutgers.edu/TechnicalReports/, 2002.

[2] H. Balakrishnan, V N. Padmanabhan, S. Seshan, and R H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[3] Y. Lu and R W. Brodersen, "Integrating power control, error correction coding, and scheduling for a CDMA downlink system," *IEEE Selected Areas in Communications*, vol. 17, no. 5, pp. 978–989, May 1999.

[4] T. Nandagopal, S. Lu, and V. Bharghavan, "A unified architecture for the design and evaluation of wireless fair queueing algorithms," in *ACM Mobicom*, 1999, pp. 132–142.

[5] S. Ramakrishna and J M. Holtzman, "A scheme for throughput maximization in a dual-class CDMA system," *IEEE Selected Areas in Communications*, vol. 16, no. 6, pp. 830–844, August 1998.

[6] N. Bambos, "Toward power-sensitive network architectures in wireless communications: concepts, issues, and design aspects," *IEEE Personal Communications*, vol. 5, no. 3, pp. 50–59, June 1998.

[7] D. Karger, C. Stein, and J. Wein, "Scheduling algorithms," in *Algorithms and Theory of Computation Handbook, CRC Press*, 1999.

[8] J. Turek, J. Wolf, and P. Yu, “Approximate algorithms for scheduling parallel tasks,” in *In Proc. of the 4th Annual Symposium on Parallel Algorithms and Architectures*, 1992, pp. 323–332.

[9] W. Ludwig and P. Tiwari, “Scheduling malleable and nonmalleable parallel tasks,” in *In Proc. of the 5th ACM SIAM Symposium on Discrete Algorithms and Architectures*, 1994, pp. 167–176.

[10] K. Jansen and L. Porkolab, “Linear-time approximation schemes for scheduling malleable parallel tasks,” in *In Proc. of the 10th ACM SIAM Symposium on Discrete Algorithms*, 1999, pp. 490–498.

[11] J. Turek, W. Ludwig, J. L. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schweigelshohn, and P. S. Yu, “Scheduling parallelizable tasks to minimize average response time,” in *In Proc. of the 6th Annual Symposium on Parallel Algorithms and Architectures*, 1994, pp. 200–209.

[12] R. Lepère, D. Trystram, and G. Woeginger, “Approximation algorithms for scheduling malleable tasks under precedence constraints,” in *Proc. of the 9th Annual European Symposium on Algorithms, LNCS 2161*, 2001, pp. 146–157.

[13] S. Acharya and S. Muthukrishnan, “Scheduling on-demand broadcasts for heterogenous: new metrics and algorithms,” in *ACM MobiCom*, 1998, pp. 43–54.

[14] N. Joshi, S R. Kadaba, S. Patel, and G. Sundaram, “Downlink scheduling in CDMA data networks,” in *ACM MobiCom*, 2000, pp. 179–190.

[15] T S. Rappaport, *Wireless Communications, Principles & Practice*, Prentice Hall, Inc., Piscataway, New Jersey, 1996.

[16] J M. Cioffi, G P. Dudevoir, M V. Eyuboglu, and G D. Forney, “MMSE decision-feedback equalizers and coding: Parts I & II,” *IEEE Transactions on Communications*, vol. 43, no. 10, pp. 2582–2604, Oct. 1995.

[17] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi, “CDMA/HDR; A bandwith-efficient high-speed wireless data service for nomadic users,” *IEEE Communications Magazine*, vol. 38, no. 7, pp. 70–77, July 2000.

[18] S. Nanda, K. Balachandran, and S. Kumar, “Adaptation techniques in wireless packet data services,” *IEEE Communications Magazine*, vol. 38, no. 1, pp. 54–65, January 2000.

[19] M. Bender, S Chakrabarti, and S. Muthukrishnan, “Flow and stretch metrics for scheduling continuous job streams,” in *In Proc. of Annual Symposium on Discrete Algorithms (SODA ’98)*, 1998, pp. 270–279.

[20] S. Chakrabarti and S. Muthukrishnan, “Resource scheduling for parallel database and scientific applications,” in *In Proc. of the ACM Symposium on Parallel Algorithms and Architectures*, 1996, pp. 329–335.

[21] H. Shachnai and J J. Turek, “Multiresource malleable task scheduling to minimize response time,” *Information Processing Letters*, vol. 70, no. 1, pp. 47–52, 1999.

[22] M R. Garey and D S. Johnson, *Computers and intractability : a guide to the theory of NP-completeness*, W H. Freeman., San Francisco,

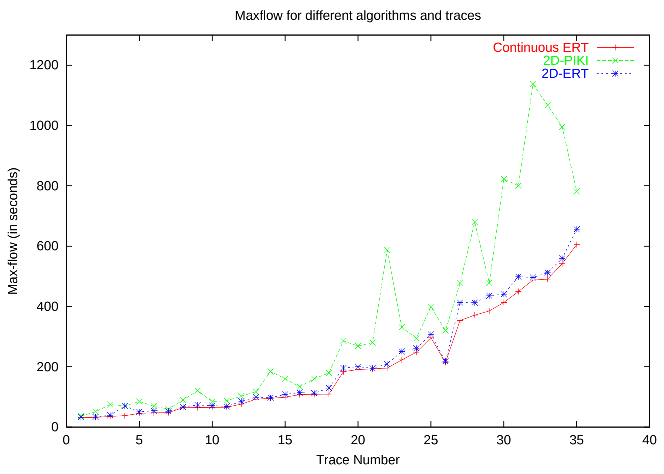

Figure 1: On-line Heuristics: Performance

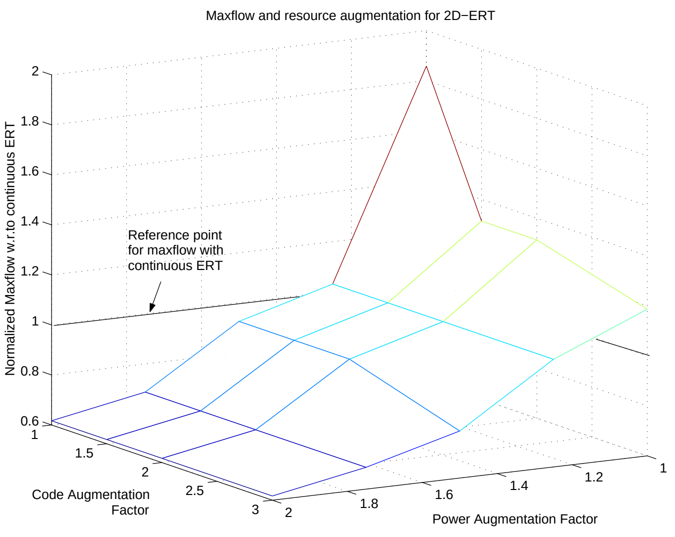

Figure 2: On-line Heuristics: Max-Flow with Resource Augmentation

1979.

[23] B. Kalyanasundaram and K.Pruhs, “Speed is as powerful as clairvoyance,” in *IEEE Symposium on Foundations of Computer Sci.*, 1995, pp. 214–221.

[24] K.R. Baker, *Introduction to Sequencing and Scheduling*, Wiley, 1974.

[25] “LOQO Optimization Toolkit,” http://www.orfe.princeton.edu/ loqo, 2000.

APPENDIX

A. PROOF OF LEMMA 4.1

Proof. For each job j of size s_j on the original instance $\mathcal{I}$, let the pair p_j^{GS}, k_j^{GS} be such that

$$\left(p_j^{GS}, k_j^{GS}\right) = \arg \min_{(p_j^c, k_j^c): s_j \le W k_j^c \log(1+\frac{g_j p_j^c}{k_j^c})} \left[\frac{p_j^c}{P} + \frac{k_j^c}{C}\right] \tag{12}$$

where p_j^{GS}, k_j^{GS} are the total power and codes assigned to user j. This solution allocates a power per code $p_j^s =$

Time Indexed Program (integral)	Interval Indexed Program (fractional)
maximize 1	maximize 1
subject to	subject to
$\sum_{j=1}^{n} c(j,t) \leq C \ \forall t \quad \sum_{j=1}^{n} p(j,t) \leq P, \quad \forall t$	$\sum_{j=1}^{n} c(j,k) \leq C \ \forall k \quad \sum_{j=1}^{n} p(j,k) \leq P, \quad \forall k$
$\sum_{t=a_j}^{d_j-1} W c(j,t) \log\left(1 + \frac{p(j,t)g_j}{c(j,t)}\right) \geq s_j, \ \forall j$	$\sum_{t=a_j}^{d_j-1} W \tau_k c(j,k) \log\left(1 + \frac{p(j,k)g_j}{c(j,k)}\right) \geq s_j, \ \forall j$
$\sum_{t<a_j, t\geq d_j} c(j,t) = 0, \ \forall j$	$\sum_{k<a_j^{-1}, k>d_j^{-1}} c(j,k) = 0, \ \forall j$
$\sum_{t<a_j, t\geq d_j} p(j,t) = 0, \ \forall j$	$\sum_{k<a_j^{-1}, k>d_j^{-1}} p(j,k) = 0, \ \forall j$
$c(j,t), p(j,t)$ discrete values	$c(j,k), p(j,k) \geq 0, \ \forall j,k$

Table 1: Offline scheduling programs

Trace	OPT	Continuous	Discrete		
	(in slots)	FIFO	FIFO	2D-FIFO	2D-PIKI
config1	109257	109891	120682	114054	114587
config2	50249	50637	55281	52263	59467
config3	36460	36725	40325	38540	46432
config4	16224	16254	17280	17210	24711

Table 2: Online Heuristics: Performance

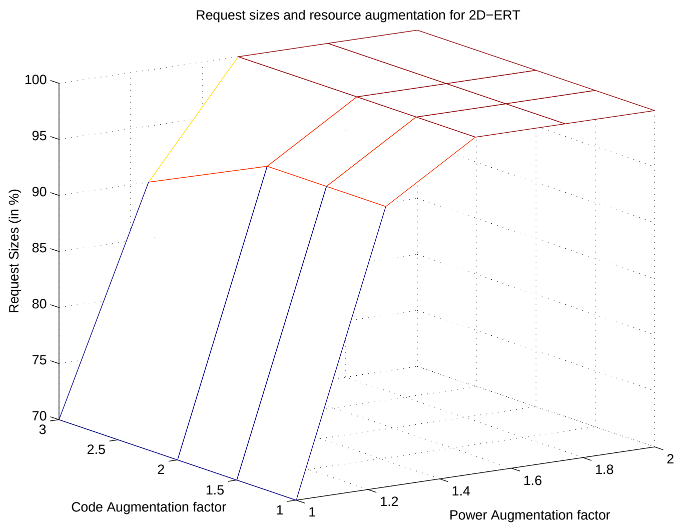

Figure 3: On-line Heuristics: Reduced Demand and Resource Augmentation

p_j^{GS}/k_j^{GS} to the jth job/user. We now show that allocating power P/C per code is not much worse in terms of this criterion. For each code which uses power p_j^s we allocate $r_j^s = \left\lceil \frac{p_j^s}{P/C} \right\rceil$ codes with power $\frac{P}{C}$ in each code. Since $r_j^s \geq \frac{p_j^s}{P/C}$,

$$s_j \overset{(a)}{\leq} W k_j^{GS} \log(1 + p_j^s g_j) \qquad \leq W k_j^{GS} \log\left(1 + r_j^s \frac{P}{C} g_j\right) \tag{13}$$

$$\leq W k_j^{GS} r_j^s \log\left(1 + \frac{P}{C} g_j\right) \qquad .$$

where (a) is due to (12). Hence using this allocation the demand s_j of each user j is satisfied. As a result we can give $k_j^{alloc} \overset{def}{=} k_j^{GS} \lceil \frac{p_j^{GS}/k_j^{GS}}{P/C} \rceil$ codes of power $\frac{P}{C}$ and still complete the job. Therefore, for the P/C allocation, if we use k_j^{alloc} codes for the job j, we obtain,

$$\frac{k_j^{alloc}}{C} = \frac{k_j^{GS}}{C} \left\lceil \frac{p_j^{GS}/k_j^{GS}}{P/C} \right\rceil \leq \left(\frac{p_j^{GS}}{P} + \frac{k_j^{GS}}{C} \right) \tag{14}$$

If p_j^{alloc} is the overall power allocated to j, we have $p_j^{alloc} = k_j^{alloc} P/C$, and hence,

$$\frac{1}{2} \left(\frac{p_j^{alloc}}{P} + \frac{k_j^{alloc}}{C} \right) = \frac{k_j^{alloc}}{C} \leq \left(\frac{p_j^{GS}}{P} + \frac{k_j^{GS}}{C} \right) \tag{15}$$

We now relate this to the optimal solution on the instance $\mathcal{I}$. Denote by $p_j(\mathcal{I})$ and $k_j(\mathcal{I})$ respectively the total power and the number of codes allocated to j when the P/C allocation is applied to instance $\mathcal{I}$. Therefore[13],

$$\frac{k_j(\mathcal{I}')}{C} \leq \frac{1}{2} \frac{k_j(\mathcal{I})}{C} \qquad \leq \frac{1}{2} \left(\frac{p_j^{GS}(\mathcal{I})}{P} + \frac{k_j^{GS}(\mathcal{I})}{C} \right) \tag{16}$$

$$\leq \frac{1}{2} \left(\frac{p_j^{OPT}(\mathcal{I})}{P} + \frac{k_j^{OPT}(\mathcal{I})}{C} \right) \quad \leq \max \left\{ \frac{p_j^{OPT}(\mathcal{I})}{P}, \frac{k_j^{OPT}(\mathcal{I})}{C} \right\}$$

$\square$

[13]Note that the optimal assignment need not necessarily assign equal power per code across the time slots where it schedules the jth user. However, due to the joint concavity of the rate in terms of power and code assignment (see in proof of Theorem 3.3) the rate for a given user can only increase by giving equal power assignment per code across time-slots, provided the power constraint is satisfied. Therefore, the optimal solution has a tighter constraint than the minimization in (12) and hence the third inequality in (16) is satisfied.