Online Multi-Resource Allocation for Deadline Sensitive Jobs with Partial Values in the Cloud

Zizhan Zheng* and Ness B. Shroff†
*Dept. of Computer Science, University of California, Davis, CA
†Dept. of ECE and CSE, The Ohio State University, Columbus, OH

***Abstract*—In many applications including interactive services and big data analytics, a timely result with a good match is often more valuable than a perfect yet delayed result. This fact can be utilized to improve the total utility gain of a cloud computing platform by allowing *partial execution* of jobs. A fundamental challenge, however, is that in many real environments, scheduling decisions have to be made online without knowledge about future jobs, which makes it difficult to choose between more valuable jobs with large deadlines and less valuable jobs that are more emergent. Moreover, jobs are often heterogeneous in their utilities, deadlines, and demands for different types of resources. In this paper, we study the problem of online scheduling for deadline-sensitive jobs with concave utility functions that can deliver partial results. We develop efficient online multi-resource allocation algorithms that achieve low competitive ratios for both continuous and discrete job models.**

I. Introduction

Cloud computing is becoming the de facto computing platform for large scale commercial applications due to its flexibility, elasticity, and cost-effectiveness. Many applications supported by the cloud, such as web search and big data analytics, are time-sensitive, where even a slight increase in delay may hurt user experience and result in revenue loss, as reported by Google, Amazon, and Microsoft [6], [27], [29]. For example, when Google displayed 30 results instead of 10, the delay grew from 400ms to 900ms, and the traffic dropped by about 20% [24]. Moreover, it is often the case that a timely result with a good match is preferable to the completed but delayed result. For instance, in a web search, returning the top few search results in a short period of time is often good enough [18]. Similarly, approximation analytics that deliveries approximate results with a delay or an error bound is starting to find various applications [1]. For these applications, *partial execution* can provide a higher utility gain than full execution, where each job has to be completely served to reap its value.

In this paper, we study the problem of scheduling deadline sensitive jobs that can deliver partial values in a cloud environment. The objective is to utilize partial execution to maximize the total utility gain from all the jobs subject to their deadline constraints. A fundamental challenge is that in many real environments, scheduling decisions have to be made online without knowledge about future job arrivals, which makes it difficult to choose between more valuable jobs (in terms of utility) with large deadlines and less valuable yet more emergent jobs that are competing for the limited computing resources. Moreover, jobs are often heterogenous in their utilities, deadlines, as resource requirements. The situation is further complicated by the fact that the utility gain of a partially finished job is often non-linear. For instance, a measurement from 200K queries in a production trace of Bing search engine [18] shows that the response quality improves with increasing resources (or time), and the relationship between the two is close to a concave function. A similar observation is made in [32] using field experiment.

This work has been supported in part by NSF grant CNS-1446582, DTRA grant HDTRA1-14-1-0058, and a grant from the Army Research Office MURI grant W911NF-12-1-0385.

An important promise of cloud computing is to enable fine grained resource sharing to make more efficient use of computing resources. Since jobs in cloud systems often require multiple types of resources such as CPU, memory, and network bandwidth, and different jobs may have very different demands for different types of resources [15], multi-resource sharing has received considerable interest in the last few years. Efficient and fair multi-resource allocation schemes have been considered in both the offline setting [15], [20], [26] and the online setting [21]. However, multi-resource allocation for deadline sensitive jobs is not well understood yet.

In this work, we develop efficient algorithms for multi-resource allocation for serving deadline sensitive jobs with partial values. We consider the online setting where jobs arrive on the fly and the scheduler has no knowledge about future arrivals. Each job is characterized by an arrival time, a deadline, a demand for each type of resource per unit of job execution, a parallelism constraint that models the maximum number of units of a job that can be executed at the same time, and a concave utility function. Jobs are assumed to be preemptive. We have designed efficient online algorithms for both continuous and discrete job models (explained below) that achieve low competitive ratios. To study their performance, we use the dual fitting technique as a principled approach.

Scheduling of deadline sensitive job have been considered in both full execution [19], [23], [25] and partial execution modes [8], [12], [18], [32]. Early works on partial execution focus on linear utility [8], [12]. Online algorithms for serving deadline sensitive jobs with non-linear utility have been recently considered in [18], [32]. In particular, a continuous job model is considered in [18] where a job can be served to any partial level, and a discrete job model is considered in [32], where each job is composed of multiple tasks of unit length. However, all these works consider a single type of resource, that is CPU. Moreover, works on partial execution all

assume that jobs can be fully parallelized. In practice, however, there is often a bound on the maximum number of tasks that can be served at the same time for any job. We extend both the continuous and the discrete job models by further considering multi-resource sharing and parallelism bounds. While focusing on concave utility functions, we expect that our approach can be combined with techniques from full execution mode [19], [25] to derive efficient solutions for jobs with more complicated utility functions, e.g., sigmoid utilities [30].

We have made following contributions in this paper.

- For the continuous job model, we show that a simple deadline oblivious algorithm that solves a convex optimization problem in each time slot achieves a small competitive ratio of 2. For the single resource setting with linear utility functions, this algorithm reduces to the FirstFit algorithm and is known to have a competitive ratio of 2 [8]. However, its performance for general concave utility functions and multiple resource types is previously unknown. Moreover, we show that this algorithm is optimal in the special case of single resource, fully parallelizable jobs with common deadlines.
- For the discrete job model, we show that if the discrete counterpart of the local convex optimization problem in the continuous case can be approximated within a factor of θ, then a simple deadline oblivious algorithm is $1+\theta$ competitive. When there is a single resource and each task requires one unit of that resource, the local problem can be solved to optimal by a simple greedy algorithm. Therefore, our theorem recovers the factor 2 result previously proved using a charging argument in this special case [17], [22]. However, our result extends to the more general heterogeneous and multi-resource setting where the local problem is NP-hard in general. For the single resource setting with parallelism constraints, we further develop a deadline aware algorithm that is 1.8-competitive.

We note that it is useful to understand the performance of deadline oblivious algorithms as they are often easier to implement than deadline aware solutions. In addition, they are applicable even when the scheduler does not have the accurate deadline information about individual jobs when they arrive. Moreover, it is worth noting that for the continuous job model with a single type of resource and linear utility functions, no online algorithms (including the deadline aware ones) can have competitive ratio less than 1.25 [11], while for the discrete model with a single type of resource, no online algorithms can achieve competitive ratio less than $\frac{\sqrt{5}+1}{2} \approx 1.618$ [17], even without parallelism constraints.

The rest of the paper is organized as follows. We introduce the system model and problem formulation in Section II. We then present the deadline oblivious online algorithm for the continuous job model and study its performance in Section III. Online algorithms for the discrete job model are discussed in Section IV. We present the numerical results in Section V, and discuss related work in Section VI. We conclude the paper in Section VII.

II. System Model and Problem Formulation

Consider a computing cloud with m types of resources, e.g., CPU, memory, and network bandwidth, shared by computing jobs. Let C_i denote the capacity of type i resource. A time-slotted system is assumed. Consider a set of computing jobs that arrive online. Each job consists of multiple tasks, and is represented by a tuple $(a_j, d_j, X_j, Y_j, f_j, \{r_{ij}\})$, where a_j and d_j denote the arrival and the deadline of job j, respectively, Y_j denotes the total number of tasks to be executed for job j, and f_j is the utility function for job j. We assume that all the tasks of job j are identical, each requiring a single unit of time, and R_{ij} units of type i resource. Let $r_{ij} \triangleq R_{ij}/C_i$ denote the proportion of resource i acquired by a task of job j. In any time slot, at most X_j tasks of job j can be run in parallel. Jobs are assumed to be preemptive. If x_j tasks of job j are finished by d_j, a value of $f_j(x_j)$ is obtained. We assume that $f_j(\cdot)$ is non-decreasing and concave, and $f_j(0) = 0$.

Let J denote the set of jobs, and $T \triangleq \max_j d_j$ the time horizon. Our objective is to maximize the total valuation from all the jobs subject to the resource constraints and the parallelism constraints in each time slot, and the deadline constraints of the jobs. Formally, we study the following optimization problem, where x_{jt} denotes the number of tasks of job j that are served at time t, and $\mathbf{x} \triangleq \{x_{jt}\}$:

$$\begin{aligned} \max_{\mathbf{x}} \quad & F(\mathbf{x}) = \sum_{j\in J} f_j(\sum_t x_{jt}) \\ \text{s.t.} \quad & \sum_t x_{jt} \le Y_j, \quad \forall j, \\ & \sum_{j\in J} r_{ij} x_{jt} \le 1, \forall i, t, \\ & x_{jt} \le X_j, \qquad \forall j, t, \\ & x_{jt} = 0, \qquad \forall j, t \notin [a_j, d_j], \\ & x_{jt} \ge 0, \qquad \forall j, t. \end{aligned} \tag{1}$$

where the first constraint indicates that there is no extra value to serve more than Y_j tasks of job j, and the second constraint is the resource capacity constraint for each type of resource. The third constraint indicates the parallelism bound for each job in each time slot, and the fourth constraint indicates that job j is not available beyond its availability window $[a_j, d_j]$.

We consider an online setting where the scheduler has no knowledge about future job arrivals. Our objective is to design online algorithms that are efficient in a provable (competitive ratio) sense. In particular, an online algorithm is q-competitive for some $q \ge 1$ if it achieves at least $1/q$ of the optimal offline value in the worst case [3].

We consider both the continuous and the discrete job models. In the continuous case, we allow x_{jt} to take continuous values in the optimization problem, and moreover, partial execution of a task provides partial value. In this case, we further assume that $f_j(\cdot)$ is continuously differentiable. On the other hand, in the discrete case, x_{jt} is constrained to take integer values. In this case, f_j can be equivalently defined by a set of marginal values $v_{jk} = f_j(k) - f_j(k-1)$. By the concavity of $f_j(\cdot)$, we have $v_{j,k+1} \le v_{j,k}, \forall j, k$.

III. Online Scheduling for Continuous Tasks

In this section, we consider the case when all x_{jt} are continuous variables. We further assume that $f_j(\cdot)$ is continuously differentiable for all j. Under these assumptions, we provide an online algorithm that is 2-competitive. We further show that the algorithm is optimal when all the jobs have a common deadline and are fully parallelizable, and there is a single type of resource (different jobs may require different amount of resource). In addition to providing useful insights, the fractional solution also serves as an upper bound to the discrete solution.

A. Deadline-Oblivious Scheduling

Consider a time slot t. Let $y_{j,t} \triangleq \sum_{\tau=1}^{t} x_{j\tau}$ denote the total number of tasks of job j that are served by time t, and let $J_t \triangleq \{j : a_j \le t \le d_j \text{ and } y_{j,t-1} < Y_j\}$ denote the set of jobs that are active at time t. In each time slot t, given the allocations made in previous time slots represented by $\{y_{j,t-1}\}$, the algorithm finds the optimal allocation to (1) up to t, by solving the following convex program.

$$\begin{aligned} \max_{\{x_{jt}: j\in J_t\}} \quad & \sum_{j\in J_t} f_j(x_{jt}+y_{j,t-1}) \qquad (2)\\ \text{s.t.} \quad & x_{jt} \le Y_j - y_{j,t-1}, \quad \forall j \in J_t,\\ & \sum_{j\in J_t} r_{ij}x_{jt} \le 1, \quad \forall i,\\ & 0 \le x_{jt} \le X_j, \quad \forall j \in J_t. \end{aligned}$$

We note that (2) is a continuous multi-dimensional knapsack problem with a concave objective function, and can be solved efficiently [4]. Below is a formal description of the online algorithm. The algorithm is deadline oblivious since the deadline information is not used in making scheduling decisions.

Algorithm 1 Online Scheduling for Continuous Tasks

$y_{j,t} \leftarrow 0, \forall j, t$;
In each time-slot t,
1: $J_t \triangleq \{j : a_j \le t \le d_j \text{ and } y_{j,t-1} < Y_j\}$;
2: $x_{jt} \leftarrow$ optimal solution to (2) given J_t and $\{y_{j,t-1}\}$;
3: $y_{j,t} \leftarrow y_{j,t-1} + x_{jt}$

B. Analysis of Algorithm 1

In this section, we study the performance of Algorithm 1. Our main result is the following theorem.

Theorem III.1. *Algorithm 1 is 2-competitive.*

To prove the theorem, we employ the dual fitting technique applied to the Lagrangian dual of the original problem. Dual fitting and the closely related primal-dual approach have been proposed as principled approaches for the design and analysis of online algorithms [7], [14], [16], [33].

We introduce dual variables α_j, β_{it}, and γ_{jt} for the first three constraints in (1). Let $\alpha \triangleq \{\alpha_j\}$, $\beta \triangleq \{\beta_{it}\}$, and $\gamma \triangleq \{\gamma_{jt}\}$. Let $\mathbf{X}$ denote the set of $\mathbf{x}$ that satisfy the last two constraints in (1). We consider the following dual function

$$\begin{aligned} G(\alpha,\beta,\gamma) &= \max_{\mathbf{x}\in\mathbf{X}} \sum_j f_j(\sum_t x_{jt}) + \sum_j \alpha_j(Y_j - \sum_t x_{jt})\\ &\quad + \sum_{i,t}\beta_{it}(1-\sum_j r_{ij}x_{jt}) + \sum_{j,t}\gamma_{jt}(X_j - x_{jt})\\ &= \max_{\mathbf{x}\in\mathbf{X}} \sum_j \Big\{ f_j(\sum_t x_{jt}) - \sum_t(\alpha_j+\gamma_{jt}+\sum_i \beta_{it}r_{ij})x_{jt}\Big\}\\ &\quad + \sum_j \alpha_j Y_j + \sum_{j,t}\gamma_{jt}X_j + \sum_{i,t}\beta_{it}\\ &= \sum_j H_j(\alpha,\beta,\gamma) + \sum_j \alpha_j Y_j + \sum_{j,t}\gamma_{jt}X_j + \sum_{i,t}\beta_{it} \end{aligned}$$

where $H_j(\alpha,\beta,\gamma) \triangleq \max_{\{x_{jt}\}\in\mathbf{X}_j} \Big\{ f_j(\sum_t x_{jt}) - \sum_t(\alpha_j + \sum_i \beta_{it}r_{ij} + \gamma_{jt})x_{jt}\Big\}$, and $\mathbf{X}_j$ is the feasible set of x_{jt} that satisfies the last two constraints in (1).

By the weak duality theorem [2], the dual function yields an upper bound on the optimal solution of the primal problem for any $\alpha_j \ge 0, \beta_{it} \ge 0, \gamma_{jt} \ge 0, \forall i, j, t$. The main idea of dual fitting is to set dual variables (α, β, γ) based on the values of the primal variables $\widetilde{\mathbf{x}}$ determined by a (deterministic) online algorithm such that $G(\alpha,\beta,\gamma) \le qF(\widetilde{\mathbf{x}})$ for some $q \ge 1$, which then implies that the online algorithm is q-competitive.

The main trick of our proof is to set the dual variables of the global problem according to the optimal dual solutions associated with the local problems solved in each time slot. This way, we have a connection between the dual solution and the online solution obtained by Algorithm 1, the latter is simply the summation of the local dual solutions over all the time slots, thanks to the convexity of the local problems. Let $\widetilde{x}_{jt}$ denote the allocation made for job j at time t by Algorithm 1, $\widetilde{y}_{j,t} = \sum_{\tau\le t}\widetilde{x}_{j\tau}$, and $\widetilde{y}_j = \widetilde{y}_{j,T}$. Recall that $\{\widetilde{x}_{jt}\}$ is the optimal solution to the convex program (2). Consider the Lagrangian function associated with the convex program at time t:

$$\begin{aligned} L(x,\mu,a,b,c) &= \sum_{j\in J_t} f_j(x_{jt}+y_{j,t-1}) + \sum_{j\in J_t}\mu_{jt}x_{jt}\\ &\quad + \sum_{j\in J_t} a_{jt}(Y_j - y_{j,t-1} - x_{jt})\\ &\quad + \sum_i b_{it}(1-\sum_j r_{ij}x_{jt})\\ &\quad + \sum_{j\in J_t} c_{jt}(X_j - x_{jt}) \end{aligned}$$

By the KKT conditions [2], there exist unique multipliers $\mu^*_{jt} \ge 0, a^*_{jt} \ge 0$, $b^*_{it} \ge 0$, and $c^*_{jt} \ge 0$, such that $\nabla_x L(\widetilde{x}, \mu^*, a^*, b^*, c^*) = 0$, $\mu^*_{jt} = 0$ if $\widetilde{x}_{jt} > 0$, $a^*_{jt} = 0$ if $\widetilde{x}_{jt} + \widetilde{y}_{j,t-1} < Y_j$, $b^*_{it} = 0$ if $\sum_j r_{ij}\widetilde{x}_{jt} < 1$, $c^*_{jt} = 0$ if $\widetilde{x}_{jt} < X_j$.

Given the solution $\{\widetilde{x}_{jt}\}$ found by Algorithm 1, we define the dual variables as

$$\begin{aligned} \alpha_j &= f_j'(\widetilde{y}_j)\mathbf{1}_{\widetilde{y}_j = Y_j}, \quad \beta_{it} = b^*_{it},\\ \gamma_{jt} &= f_j'(\widetilde{y}_j)\mathbf{1}_{\widetilde{x}_{jt}=X_j \text{ and } \widetilde{y}_j < Y_j} \end{aligned}$$

It is clear that $\alpha_j \ge 0$, $\beta_{it} \ge 0$, $\gamma_{jt} \ge 0, \forall i, j, t$. Moreover, we have the following properties.

Lemma III.1. $H_j(\alpha,\beta,\gamma) \le f_j(\widetilde{y}_j) - f_j'(\widetilde{y}_j)\widetilde{y}_j$.

Proof. We first claim that for any $t \in [a_j, d_j]$, $\alpha_j + \sum_i \beta_{it} r_{ij} + \gamma_{jt} \geq f_j'(\widetilde{y}_j)$. If $\widetilde{y}_j = Y_j$ or $\widetilde{y}_j < Y_j$ and $\widetilde{x}_{jt} = X_j$, then $\alpha_j + \gamma_{yt} = f_j'(\widetilde{y}_j)$ by definition and the claim is clearly true. Suppose $\widetilde{y}_j < Y_j$ and $\widetilde{x}_{jt} < X_j$. Then $\alpha_j = \gamma_{jt} = 0$. By the KKT conditions, we have $a_{jt}^* = 0$ since $\widetilde{x}_{jt} + \widetilde{y}_{j,t-1} \leq \widetilde{y}_j < Y_j$, and $c_{jt}^* = 0$ since $\widetilde{x}_{jt} < X_j$. Moreover,

$$\begin{aligned} 0 &= \frac{\partial L}{\partial x_{jt}}(\widetilde{x}, \mu^*, a^*, b^*, c^*) \\ &= f_j'(\widetilde{x}_{jt} + \widetilde{y}_{j,t-1}) + \mu_{jt}^* - a_{jt}^* - \sum_i b_{it}^* r_{ij} - c_{jt}^* \qquad (3) \\ &\overset{(a)}{\geq} f_j'(\widetilde{x}_{jt} + \widetilde{y}_{j,t-1}) - \sum_i \beta_{it} r_{ij} \\ &\overset{(b)}{\geq} f_j'(\widetilde{y}_j) - \sum_i \beta_{it} r_{ij} \end{aligned}$$

where (a) follows from $\mu_{jt}^* \geq 0$ and $a_{jt}^* = c_{jt}^* = 0$, and (b) follows from the concavity of $f_j(\cdot)$. The claim then follows. Let $t(j) \triangleq \text{argmin}_{t \in [a_j, b_j]}(\alpha_j + \sum_i \beta_{it} r_{ij} + \gamma_{jt})$. We have

$$\begin{aligned} H_j(\alpha, \beta, \gamma) &= \max_{y_j \geq 0} f_j(y_j) - (\alpha_j + \sum_i \beta_{it(j)} r_{ij} + \gamma_{jt(j)}) y_j. \\ &\leq \max_{y_j \geq 0} f_j(y_j) - f_j'(\widetilde{y}_j) y_j = f_j(\widetilde{y}_j) - f_j'(\widetilde{y}_j) \widetilde{y}_j \end{aligned}$$

□

Lemma III.2. *For any t, $\sum_i \beta_{it} \leq \sum_j f_j'(\widetilde{y}_{jt}) \widetilde{x}_{jt}$.*

Proof. Since $\beta_{it} = b_{it}^* = 0$ if $\sum_j r_{ij} \widetilde{x}_{jt} < 1$, we have

$$\begin{aligned} \sum_i \beta_{it} &= \sum_i \beta_{it} \sum_j r_{ij} \widetilde{x}_{jt} \\ &= \sum_j \sum_i b_{it}^* r_{ij} \widetilde{x}_{jt} \\ &\overset{(a)}{\leq} \sum_j f_j'(\widetilde{y}_{jt}) \widetilde{x}_{jt} \end{aligned}$$

where (a) follows from (3) and $\mu_{jt}^* = 0$ when $\widetilde{x}_{jt} > 0$. □

Proof of Theorem 1: From Lemmas III.1 and III.2, we have

$$\begin{aligned} &G(\alpha, \beta, \gamma) \\ &= \sum_j H_j(\alpha, \beta, \gamma) + \sum_j \alpha_j Y_j + \sum_{i,t} \beta_{it} + \sum_{j,t} \gamma_{jt} X_j \\ &\leq \sum_j (f_j(\widetilde{y}_j) - f_j'(\widetilde{y}_j) \widetilde{y}_j) + \sum_j \alpha_j Y_j + \sum_{i,t} \beta_{it} + \sum_{j,t} \gamma_{jt} X_j \\ &\overset{(a)}{=} \sum_j f_j(\widetilde{y}_j) + \sum_{j,t} \Big(\alpha_j - f_j'(\widetilde{y}_j) + \gamma_{jt}\Big) \widetilde{x}_{jt} + \sum_{i,t} \beta_{it} \\ &\overset{(b)}{\leq} \sum_j f_j(\widetilde{y}_j) + \sum_{j,t} f_j'(\widetilde{y}_{jt}) \widetilde{x}_{jt} \\ &\overset{(c)}{\leq} \sum_j f_j(\widetilde{y}_j) + \sum_j f_j(\widetilde{y}_j) = 2 \sum_j f_j(\widetilde{y}_j) \end{aligned}$$

where (a) follows from the fact that $\alpha_j = 0$ if $\widetilde{y}_j < Y_j$ and $\gamma_{jt} = 0$ if $\widetilde{x}_{jt} < X_j$, (b) follows from Lemma III.2 and the fact that $\alpha_j + \gamma_{jt} \leq f_j'(\widetilde{y}_j)$, and (c) follows from the concavity of $f_j(\cdot)$ (see Figure 1 for an explanation).

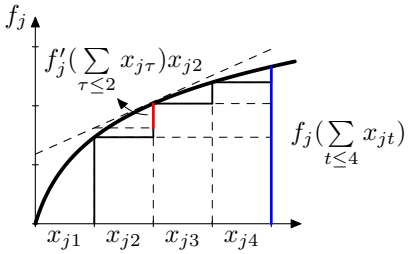


Fig. 1: An example that shows $\sum_j f_j'(\sum_{\tau \leq t} x_{j\tau}) x_{jt} \leq f_j(\sum_t x_{jt})$.

Remark 1: When $f_j(\cdot)$ is a linear function for all j, i.e., $f_j(y_j) \triangleq v_j y_j$ for some $v_j > 0$ for all j, and there is a single type of resource, say i, and $R_{ij} = 1$ for all j, Algorithm 1 reduces to the simple FirstFit algorithm, which is known to be 2-competitive (and this bound is tight) [8]. However, its performance for general concave valuations and multiple resource types is *previously unknown*.

C. Jobs with Common Deadlines

We now consider the special case when all the jobs have the same deadline T. We show that Algorithm 1 is optimal when there is a single type of resource and there is no parallelism constraint. However, optimality is not guaranteed if either of the two conditions does not hold as we show below.

Example 1 (common deadline and multiple resources): Consider two types of resources, and three jobs where $a_1 = 1, d_1 = 2, X_1 = Y_1 = 1, f_1(x) = 2x, r_{11} = 0.2, r_{21} = 0.8$, $a_2 = 1, d_2 = 2, X_2 = Y_2 = 2, f_2(x) = x, r_{12} = 0.5, r_{22} = 0.5$, and $a_3 = 2, d_3 = 2, X_3 = Y_3 = 2, f_3(x) = 2x, r_{13} = 0.8, r_{23} = 0.2$. By solving the LPs, Algorithm 1 serves 1 unit of job 1 and 0.4 unit of job 2 in the first time slot and 1.25 units of job 3 in the second time slot, and has a total valuation of $f_1(1) + f_2(0.4) + f_3(1.25) = 4.9$. On the other hand, a schedule that serves 2 units of job 2 in the first time slot, and 1 unit of job 1 and 1 unit of job 3 in the second time slot is feasible, and has a higher valuation of $f_1(1) + f_2(2) + f_3(1) = 6$.

Example 2 (common deadline and parallelism constraint): Consider two jobs and a single type of resource, where for both jobs, $a_j = 1, d_j = 2, X_j = 0.75$, and $r_j = 1$. $Y_1 = 0.5, Y_2 = 2$. Both jobs have the same f_j that is strictly concave. Then Algorithm 1 serves 0.5 units of both jobs at time 1, and 0.75 unit of job 2 at time 2 due to the parallelism constraint. However, an optimal schedule serves 0.25 unit of job 1 and 0.75 unit of job 2 in each time slot, and obtains a higher utility when f_j is strictly increasing.

Before we prove Algorithm 1's optimality for the common deadline case, we first establish its following properties. For any time slot t, we again let J_t denote the set of active jobs at t. Let $J_t^1 \subseteq J(t)$ denote the subset of jobs with $\widetilde{x}_{jt} > 0$, and $J_t^2 \subseteq J_t^1$ the subset of J_t^1 with $\widetilde{y}_{j,t} < Y_j$.

Lemma III.3. *In Algorithm 1, for any jobs in J_t^2, $f_j'(\widetilde{y}_{j,t})/r_j$ are the same.*

Proof. For any jobs in J_t^2, we have $\mu_{jt}^* = 0$ and $a_{jt}^* = 0$ by the KKT conditions. From (3), we then have $f_j'(\widetilde{y}_{j,t}) = b_t^* r_j$ for any j (we omit index i since there is only one type of resource). The statement then follows. □

It follows that, without loss of optimality, we can assume that if two jobs start with the same ratio of $f_j'(\widetilde{y}_{j,t})/r_j$ in the beginning of time t, then using Algorithm 1 either both of them are served or non of them are served in t, and they end up with the same ratio at the end of t, unless one of them is fully served during t. By induction and the common deadline assumption, we then have the following result:

Lemma III.4. *Suppose $j_1 \in J_t^1$ and $j_2 \in J_t^1$. Then (1) for any time slot $t' > t$ where $j_1 \in J_{t'}^1$, we must have $j_2 \in J_{t'}^1$ unless $\widetilde{y}_{j_2,t'-1} = Y_{j_2}$; (2) $f_{j_1}'(\widetilde{y}_{j_1})/r_{j_1} = f_{j_2}'(\widetilde{y}_{j_2})/r_{j_2}$ if $\widetilde{y}_{j_1} < Y_{j_1}$ and $\widetilde{y}_{j_2} < Y_{j_2}$.*

To establish the optimality of the algorithm, we redefine β_t (we omit the resource index i for the single resource case) as follows. For any time slot t, we set $\beta_t = 0$ if $\sum_j r_j \widetilde{x}_{jt} < 1$. Otherwise, we let $t' \geq t$ denote the last time slot where $x_{jt'} > 0$ for some $j \in J_t$, and set $\beta_t = \min_{j \in J_t : x_{jt'} > 0} f'(\widetilde{y}_j)/r_j$. We then set $\alpha_j = f_j'(\widetilde{y}_j) - \min_{t \geq a_j} \beta_t r_j$ if $\widetilde{y}_j = Y_j$, and $\alpha_j = 0$ otherwise.

Lemma III.5. *$\alpha_j \geq 0$ for all j.*

Proof. The statement is clearly true when $\widetilde{y}_j < Y_j$. Assume $\widetilde{y}_j = Y_j$. Let t denote the last time slot where $\widetilde{x}_{jt} > 0$. We either have $\beta_t = 0$, or $\beta_t = f_j'(\widetilde{y}_j)/r_j$, or $\beta_t = f_{j'}'(\widetilde{y}_{j'})/r_{j'}$ for some j' with $\widetilde{x}_{j't} > 0$. In the last case, we must have $f_{j'}'(\widetilde{y}_{j'})/r_{j'} \leq f_j'(\widetilde{y}_j)/r_j$ by the greedy way that jobs are served, which implies the statement. □

Lemma III.6. *For any job j and time slot $t \geq a_j$, (1) $f_j'(\widetilde{y}_j) - \alpha_j - \beta_t r_j \leq 0$ if $\widetilde{x}_{jt} = 0$ and (2) $f_j'(\widetilde{y}_j) - \alpha_j - \beta_t r_j = 0$ if $\widetilde{x}_{jt} > 0$.*

Proof. To show the first property, consider any time slot $t \geq a_j$ with $\widetilde{x}_{jt} = 0$. First assume $\widetilde{y}_j < Y_j$. Then $\alpha_j = 0$. Moreover, we must have $\beta_t \geq f_j'(\widetilde{y}_j)/r_j$ by the definition of β_t. On the other hand, when $\widetilde{y}_j = Y_j$, the claim follows directly from the definition of α_j.

We then prove the second property. Consider any time slot $t \geq a_j$ with $\widetilde{x}_{jt} > 0$. First assume $\widetilde{y}_j < Y_j$. Then $\alpha_j = 0$. Moreover, by the the definition of β_t and the common deadline assumption, we must have $\beta_t = f_j'(\widetilde{y}_j)/r_j$. On the other hand, when $\widetilde{y}_j = Y_j$, we either have $\beta_t = 0$, or $\beta_t = \min_{t' \geq a_j} \beta_{t'}$. In both cases, the second property holds. □

Theorem III.2. *Algorithm 1 is optimal when all the jobs have the same deadline and share a single type of resource.*

Proof. We define $G(\alpha, \beta)$ and $H(\alpha, \beta)$ similar to $G(\alpha, \beta, \gamma)$ and $H(\alpha, \beta, \gamma)$ with the parallelism constraints removed. Using Lemma III.6, we can show that $H(\alpha, \beta)$ satisfies the same condition in Lemma III.1. We then have $G(\alpha, \beta) \leq \sum_j f_j(\widetilde{y}_j) + \sum_t (\beta_t r_j + \alpha_j - f_j'(\widetilde{y}_j))\widetilde{x}_{jt} = \sum_j f_j(\widetilde{y}_j)$ by the second property of Lemma III.6. □

IV. Online Scheduling for Discrete Tasks

In this section, we study the discrete setting where x_{jt} are constrained to be integers. We first show that a deadline oblivious algorithm similar to Algorithm 1 again provides a small approximation factor as long as the discrete counterpart of the local problem can be well approximated. To prove this result, we reformulate the problem by taking a task view, and utilize Fenchel duality [2] that is more suitable to our dual fitting argument in the discrete case.

Algorithm 2 Online Scheduling for Discrete Tasks

$x_{st} \leftarrow 0, \forall s, t$;
In each time-slot t,
1: $S_t \leftarrow \{s \in S : a_s \leq t \leq d_s, x_{st'} = 0 \text{ for } t' < t\}$;
2: $x_{st} \leftarrow$ apply ALGO to find a solution to (5) on S_t

A. Deadline-Oblivious Scheduling

We first observe that in the discrete case, the original problem (1) can be converted into the following equivalent form. For each job j, we create Y_j tasks where the k-th task takes a value equal to the k-th marginal valuation of the job, determined by $f_j(\cdot)$, and all the tasks of job j have the same arrival time and deadline, and the same resource requirement. Let $a_s, d_s, v_s, \{r_{is}\}$ denote the arrival time, deadline, the value, and the resource requirement of a task s. We then treat each task as a single unit job. Let S_j denote the set of tasks of job j, and let $S = \bigcup_j S_j$ be the entire set of tasks from all the jobs. We can then rewrite problem (1) by taking the task view as follows, where x_{st} is the binary decision variable that denotes whether the task s is scheduled in time t or not.

$$\begin{aligned} \max_{\mathbf{x}} \quad & F(\mathbf{x}) = \sum_{s \in S} v_s \Big(\sum_t x_{st}\Big) && (4) \\ \text{s.t.} \quad & \sum_t x_{st} \leq 1, \quad \forall s, \\ & \sum_{s \in S} r_{is} x_{st} \leq 1, \forall i, t, \\ & \sum_{s \in S_j} x_{st} \leq X_j, \forall j, t, \\ & x_{st} \in \{0, 1\} \quad \forall j, t, \\ & x_{st} = 0, \qquad \forall j, t \notin [a_j, d_j]. \end{aligned}$$

Note that we have ignored the precedence constraint in the above formulation, which requires that k-th task of job j to be scheduled before its $k+1$-th task. This is without loss of optimality due to the concavity of $f_j(\cdot)$. Similar to the continuous case, we consider a deadline-oblivious algorithm that solves in each time slot the following discrete counterpart of problem (2), where $S_t \triangleq \{s \in S : a_s \leq t \leq d_s, x_{st'} = 0 \text{ for } t' < t\}$ denotes the set of active tasks at time t.

$$\begin{aligned} \max_{\{x_{st} : s \in S_t\}} \quad & \sum_{s \in S_t} v_s x_{st} && (5) \\ \text{s.t.} \quad & \sum_{s \in S_t} r_{is} x_{st} \leq 1, \forall i, \\ & \sum_{s \in S_j} x_{st} \leq X_j, \ \forall j, \\ & x_{st} \in \{0, 1\} \quad \forall s. \end{aligned}$$

Our deadline oblivious online algorithm for the discrete case is given in Algorithm 2, where ALGO is an arbitrary algorithm that solves the local problem.

B. Analysis of Algorithm 2

We again use the dual fitting technique to study the performance of Algorithm 2. Instead of Lagrangian duality, we consider Fenchel duality [2], which is more suitable to the task view of the problem in the discrete setting. Consider a general optimization problem

$$\begin{aligned} \max \quad & f_1(x) - f_2(x) \\ \text{s.t.} \quad & x \in X_1 \cap X_2 \end{aligned}$$

where f_1 and f_2 are real-valued functions on $\mathbb{R}^n$, X_1 and X_2 are subsets of $\mathbb{R}^n$. The dual problem is defined as follows:

$$\begin{aligned} \min \quad & q(\lambda) = g_2(\lambda) - g_1(\lambda) \\ \text{s.t.} \quad & \lambda \in \Lambda_1 \cap \Lambda_2, \end{aligned}$$

where

$$g_1(\lambda) = \inf_{x \in X_1} \{x'\lambda - f_1(x)\}, \;\; g_2(\lambda) = \sup_{x \in X_2} \{x'\lambda - f_2(x)\},$$
$$\Lambda_1 = \{\lambda : g_1(\lambda) > -\infty\}, \;\; \Lambda_2 = \{\lambda : g_2(\lambda) < \infty\}.$$

We again have the weak duality property, that is, any feasible dual solution provides an upper bound to the primal optimal. To derive the Fenchel dual of our problem, we introduce dual variables λ_{st} for each task s and time slot t, and set $f_1(x) = \sum_{s \in S} v_s(\sum_t x_{st})$, $f_2(x) = 0$, and $X_1 = X_2 = \mathbf{X}$ including all feasible x_{st} in (4). We then have the following dual function:

$$\begin{aligned} q(\lambda) &= g_2(\lambda) - g_1(\lambda) \\ &= \max_{\mathbf{x} \in \mathbf{X}} \Big\{ \sum_s v_s \Big(\sum_t x_{st} \Big) - \sum_{s,t} \lambda_{st} x_{st} \Big\} + \max_{\mathbf{x} \in \mathbf{X}} \sum_{s,t} \lambda_{st} x_{st} \end{aligned} \tag{6}$$

where we only consider the set of λ such that $g_1(\lambda)$ and $g_1(\lambda)$ can be achieved at feasible x_{st}. In the following, we always set dual variables such that $\lambda_{st} = \lambda_s$ for all t. In this case, the dual function can be further simplified as

$$\begin{aligned} q(\lambda) &= \max_{\mathbf{x} \in \mathbf{X}} \Big\{ \sum_s v_s \Big(\sum_t x_{st} \Big) - \lambda_s \sum_{s,t} x_{st} \Big\} + \max_{\mathbf{x} \in \mathbf{X}} \sum_{s,t} \lambda_s x_{st} \\ &= \max_{\mathbf{x} \in \mathbf{X}} \Big\{ \sum_s (v_s - \lambda_s) \sum_t x_{st} \Big\} + \max_{\mathbf{x} \in \mathbf{X}} \sum_{s,t} \lambda_s x_{st} \\ &\le \sum_{s: v_s > \lambda_s} (v_s - \lambda_s) + \max_{\mathbf{x} \in \mathbf{X}} \sum_{s,t} \lambda_s x_{st} \end{aligned} \tag{7}$$

Using the dual function, we can prove the following performance bound for Algorithm 2.

Theorem IV.1. *If in each time slot, ALGO finds a solution that is within a factor $\theta \ge 1$ of the optimal solution to the local problem, then Algorithm 2 is $1 + \theta$ competitive.*

Proof. Given the online solution $\widetilde{x}_{st}$ found by Algorithm 2, we set the dual variables as follows:

$$\lambda_s = \begin{cases} 0 & \text{if } \sum_t \widetilde{x}_{st} = 1, \\ v_s & \text{if } \sum_t \widetilde{x}_{st} = 0. \end{cases}$$

Let $V \triangleq \sum_s v_s \sum_t \widetilde{x}_{st}$ denote the value of the online solution found by Algorithm 2. We first show that the first part of $q(\lambda)$ is upper bounded by V. From the definition of λ_s, we have $v_s > \lambda_s$ iff $\sum_t \widetilde{x}_{st} = 1$. It follows that $\sum_{s: v_s > \lambda_s} (v_s - \lambda_s) \le \sum_{s: v_s > \lambda_s} v_s \le \sum_s v_s \sum_t \widetilde{x}_{st} = V$.

We then show that the second part of $q(\lambda)$ is upper bounded by θV. To see this, let $\mathbf{X}' \supseteq \mathbf{X}$ denote the set of x_{st} that may violate the first constraint in (4) while still satisfying the other constraints, and $V_t \triangleq \sum_s v_s \widetilde{x}_{st}$ the total value obtained by the algorithm at time t. We then observe that

$$\begin{aligned} \max_{x \in \mathbf{X}} \sum_{s,t} \lambda_s x_{st} &\le \max_{x \in \mathbf{X}'} \sum_{s,t} \lambda_s x_{st} \\ &\overset{(a)}{\le} \sum_t \max_{x \in \mathbf{X}'} \sum_s \lambda_s x_{st} \\ &\overset{(b)}{=} \sum_t \max_{x \in \mathbf{X}'} \sum_{s: \lambda_s > 0} v_s x_{st} \\ &\overset{(c)}{\le} \sum_t \max_{x \in \mathbf{X}'} \sum_{s \in S_t} v_s x_{st} \\ &\overset{(d)}{\le} \sum_t \theta V_t = \theta V, \end{aligned}$$

where (a) follows from the fact that in problem (4), all the constraints except the first one are separable over t, (b) follows from the fact that there is no benefit to consider any task with $\lambda_s = 0$, (c) follows from that for any task s with $a_s \le t \le d_s$, if $s \notin S_t$, then s has been scheduled before t in Algorithm 2, hence $\lambda_s = 0$, and (d) follows directly by comparing the separated dual problem with the local problem (5). The theorem then follows by combining the first part and the second part. □

Remark 1: When there is a single type of resource and each task requires one unit of that resource, Algorithm 2 is 2-competitive, since in this case, the local problem (5) can be easily solved by a simple greedy algorithm that always serves active tasks with highest values subject to the parallelism constraint. Thus, our dual fitting argument provides a new proof to the factor 2 result previously proved by a charging argument in [17] and [22], for the single resource setting with unit demand per task. Note that the factor 2 is tight for the greedy algorithm, which can be shown by simple examples.

Remark 2: For the general multi-resource setting, the local problem at each time slot can be viewed as an integral multi-dimensional knapsack problem. This problem is hard to approximate within a factor of $\Omega(m^{\frac{1}{B+1} - \epsilon})$ for every fixed B unless $NP = ZPP$ [9], where $B = 1/\max_{i,j} r_{ij}$. Thus, the competitive ratio of any online algorithm is likely to depend on m, the number of resource types. On the other hand, in practice, the pool of a certain type of resource in a data center is usually substantially larger than the amount required by any single task [28], [31]. Hence, we expect that $B \to \infty$. We then discuss two algorithms to the local problem with different performance guarantees.

A local greedy algorithm: We can consider a direct extension of the single resource greedy algorithm to the multi-resource setting. Let $r_s \triangleq \max_i r_{is}$ denote the amount of *dominant*

resource [15] required by task s. The algorithm first sorts the set of active tasks by v_s/r_s, and picks the set of tasks with the highest ratios subject to the resource budget and the parallelism constraint. The total value of all selected tasks is then compared with the value of the next task on the list, and the larger of the two is taken as the final solution to the local problem. The algorithm has a complexity of $O(n_t \log n_t)$ where n_t is the number of active tasks at time t. Moreover, it can be shown that the algorithm achieves an approximation factor of $\min(2, \frac{B}{B-1})m$, which approaches to m when $B \to \infty$. This result can again be proved using a dual fitting argument by considering the Lagrangian dual of the local problem.

A local primal-dual algorithm: We can also apply the primal-dual algorithm proposed in [5] to the local problem. The main idea is to view the Lagrangian dual variable b_i associated with the first constraint in (5) as the unit price for using resource i. Initially, $b_i = 1$ for all i. In each round, the algorithm picks a task s with the highest ratio of $v_s/\sum_i r_{is}b_i$, and updates $b_i = b_i(e^{B-1}m)^{\frac{r_{is}}{1-\max_{s'} r_{is'}}}$. The algorithm stops when either all the active tasks are scheduled or $\sum_i b_i > \mathbf{e}^{B-1}m$, where $\mathbf{e}$ is the base of the natural logarithm. This algorithm has a complexity of $O(n_t^2)$, and an approximation factor of $\frac{\mathbf{e}B}{B-1}m^{\frac{1}{B-1}}$, which approaches to $\mathbf{e}$ when $B \to \infty$. Thus, this algorithms works better than the greedy algorithm for large m and large B.

C. Deadline Aware Scheduling

In this section, we study the benefit of deadline awareness in the design of partial scheduling algorithms. For simplicity, we focus on the single resource setting where each task requires a single unit of that resource, while keeping the parallelism constraint. Let C denote the total units of the available resource. In this case, Algorithm 2 is 2-competitive by always picking the set of active tasks with highest values (subject to the parallelism constraint) in each time slot. The main inefficiency of this myopic strategy is that it can potentially miss the set of tasks that are slightly less valuable but are more emergent than the scheduled ones. The main idea of the deadline aware algorithm is to schedule a few tasks with small deadlines in addition to those tasks with high values in each time slot. This idea has been applied before in continuous job scheduling with linear utility [12] and unit job scheduling [10]. However, none of these works consider jobs with parallel tasks subject to a parallelism constraint. Our algorithm can be viewed as a discrete counterpart of the algorithm in [12] with the parallelism constraint taken into account.

The algorithm works as follows (see Algorithm 3). Consider a time slot t. Recall that S_t is the set of active tasks at t. From the concavity of job values, among the set of k remaining tasks of job j, it is sufficient to consider the first $\min(X_j, k)$ tasks with the highest values. Let S'_t denote these tasks from all the active jobs. These tasks are first sorted by their values non-increasingly (line 2). Let v_h denote the value of the h-th task on the list. Let S_t^p denote the first p tasks on the list, and $\bar{v}_p$ the average value of these tasks. Let $\overline{S}_t(p,\rho) \subseteq S'_t \backslash S_t^p$ denote the set of remaining tasks with values no less than $\rho\bar{v}_p$ where ρ is a parameter to be determined. We note that the greedy solution simply picks the first $\min\{|S'_t|, C\}$ tasks on the list. In contrast, our algorithm picks all these tasks only if there is no extra task available (that is, $|S'_t| \le C$), or $v_{C+1} \le \rho\bar{v}_p$, that is when the value of the $C+1$-th task on the list is significantly less than the average value of the first p tasks (line 3-4). Otherwise, the algorithm picks the first p tasks of highest values, and use the remaining resource to serve the $C-p$ tasks in $\overline{S}_t(p,\rho)$ with the minimum deadlines, where p is again a parameter to be determined (line 5-6).

Algorithm 3 Deadline Aware Scheduling for Discrete Tasks

In each time-slot t,

1: $S'_t \leftarrow$ set of active tasks that can be scheduled at t;
2: Sort the tasks in S'_t by task values non-increasingly;
3: **if** $|S'_t| \le C$ or $v_{C+1} \le \rho\bar{v}_p$ **then**
4: Schedule the first $\min\{|S'_t|, C\}$ tasks on the list;
5: **else**
6: Schedule the first p tasks on the list and the $C-p$ tasks in $\overline{S}_t(p,\rho)$ with the minimum deadlines

Note that our algorithm retains the $O(n_t \log n_t)$ complexity in each time slot, where n_t is the number of active tasks in time slot t. Below we prove that Algorithm 3 is 1.8-competitive, where we also determine the values of ρ and p.

Theorem IV.2. *Algorithm 3 is 1.8-competitive by taking $\rho = 2/3$ and $p = C/2$.*

Proof. Let A_t denote the set of tasks scheduled at time t in Algorithm 3. Let $A_t^1 = A_t \cap S_t^p$ and $A_t^2 = A_t \backslash A_t^1$. Let $M_t \subseteq A_t$ denote the set of scheduled tasks where the corresponding job achieves the maximum parallelism at t. In other words, any $s \in M_t$ belongs to a job j such that X_j tasks of j are scheduled at time t. For any active task $s \in S_t \backslash A_t$, there are three possibilities: (1) $s \in S_t^C \backslash S_t^p$, then we have $d_s \ge d_{s'}$ for any task $s' \in A_t^2$; (2) $s \in S'_t \backslash S_t^C$, then either $v_s < \rho\bar{v}_p$ or $d_s \ge d_{s'}$ for any task $s' \in A_t^2$; (3) $s \in S_t \backslash S'_t$.

We again adopt a dual fitting argument. Given the solution found by Algorithm 3, we set the dual variables λ as follows. If task s is not scheduled by its deadline, $\lambda_s = v_s$. For the set of scheduled tasks, their λ are determined in a recursive way starting from T. Consider any time slot t, and suppose the value of λ has been determined for any tasks scheduled after t. For $s \in A_t^1 \backslash M_t$, $\lambda_s = 0$. Let λ'_t denote the maximum λ value for any task in $\overline{S}_t(p,\rho) \backslash A_t$, and $\lambda'_t = 0$ if $\overline{S}_t(p,\rho) \backslash A_t = \emptyset$. For $s \in A_t^2 \backslash M_t$, $\lambda_s = \min(v_s, \lambda'_t)$. For $s \in A_t^1 \cap M_t$, $\lambda_s = \lambda_{s_t^j}$, where j is the job that s belongs to and s_t^j is the first task of j that is not scheduled at time t. For $s \in A_t^2 \cap M_t$, $\lambda_s = \min(v_s, \max(\lambda'_t, \lambda_{s_t^j}))$. Let $\lambda_t \triangleq \max_{s \in S'_t \backslash M_t} \lambda_s$.

Let $A = \bigcup_t A_t$ denote the set of tasks scheduled by Algorithm 3. Let $V_t = \sum_{s \in A_t} v_s$ denote the total value obtained at time t in Algorithm 3, and $V = \bigcup_t V_t$. For the single resource unit demand setting and from the definition of dual variables, (7) can be simplified as follows:

$$
\begin{aligned}
q(\lambda) &\le \sum_{s: v_s > \lambda_s} (v_s - \lambda_s) + \max_{\mathbf{x} \in \mathbf{X}} \sum_{s,t} \lambda_s x_{st} \\
&\le \sum_{s \in A} v_s - \sum_{s \in A} \lambda_s + \sum_t \Big(\lambda_t |A_t \backslash M_t| + \sum_{s \in M_t} \max(\lambda_t, \lambda_s) \Big)
\end{aligned}
$$

$$\leq V + \sum_t \Big(\sum_{s \in A_t^2 : \lambda_t \geq \lambda_s} (\lambda_t - \lambda_s) + \sum_{s \in A_t^1} \lambda_t \Big)$$

Let $g_t \triangleq \sum_{s \in A_t^2 : \lambda_t \geq \lambda_s} (\lambda_t - \lambda_s) + \sum_{s \in A_t^1} \lambda_t$. We distinguish the following cases:

1) $|S_t'| \leq C$: we have $\lambda_t = 0$ and $g_t = 0$.
2) $|S_t'| > C$ and $v_{C+1} \leq \rho \bar{v}_p$: we have $\lambda_t \leq v_{C+1} \leq \rho \bar{v}_p$. Hence, $\frac{g_t}{V_t} \leq \frac{(C-p)\lambda_t + p\rho\bar{v}_p}{\sum_{s \in A_t^2} v_s + p\bar{v}_p} \leq \frac{(C-p)\lambda_t + p\rho\bar{v}_p}{(C-p)\lambda_t + p\bar{v}_p} \leq \frac{(C-p)\rho + p\rho}{(C-p)\rho + p} = \frac{\rho C}{(C-p)\rho + p}$.
3) $|S_t'| > C$, $v_{C+1} > \rho \bar{v}_p$, and $\lambda_t = \lambda_{s'}$ for some $s' \notin S_t^C \cup \overline{S}_t(p, \rho)$: we have $\lambda_t \leq v_{s'} \leq \rho \bar{v}_p$, $g_t \leq \rho \bar{v}_p C$, $V_t = p\bar{v}_p + \sum_{s \in A_t^2} v_s \geq p\bar{v}_p + (C-p)\rho\bar{v}_p$. Hence, $\frac{g_t}{V_t} \leq \frac{\rho C}{p + (C-p)\rho}$.
4) $|S_t'| > C$, $v_{C+1} > \rho \bar{v}_p$, and $\lambda_t = \lambda_{s'}$ for some $s' \in S_t^C \cup \overline{S}_t(p, \rho)$: we have $\lambda_s \geq \min\{v_s, \lambda_t\}$ for any $s \in A_t^2$, and $g_t \leq C\bar{v}_p - (C-p)\rho\bar{v}_p$, and $V_t = p\bar{v}_p + \sum_{s \in A_t^2} v_s \geq p\bar{v}_p + (C-p)\rho\bar{v}_p$. Hence, $\frac{g_t}{V_t} \leq \frac{C - (C-p)\rho}{p + (C-p)\rho}$.

We then find ρ and p that minimize the worst-case g_t/V_t among the above four cases. Let $\delta_1 = \frac{\rho C}{p + (C-p)\rho}$ and $\delta_2 = \frac{C - (C-p)\rho}{p + (C-p)\rho}$. Note that $\delta_1 = \delta_2$ if $\rho C = C - (C-p)\rho$, that is when $\rho = \frac{C}{2C - p}$. Moreover, for this ρ, δ_1 (and δ_2) is minimized at $p = C/2$. Taking $p = C/2$ and $\rho = \frac{C}{2C-p} = 2/3$, we have $\delta_1 = \delta_2 = 1.8$. It follows that $q(\lambda) \leq V + \sum_t 0.8 V_t = 1.8V$. □

V. Numerical Results

In this section, we study the performance of our algorithms with numerical results. For both continuous and discrete job models, our algorithms achieve clearly higher utility gains compared with commonly adopted heuristics that ignore the heterogeneous multi-resource requirements.

Setup: We consider a data center with 100 units of CPU and 100 units of memory resources. 100 jobs are generated. The number of job arrivals in each time slot follows a Poisson distribution with a mean of 10, independent of other time-slots. The availability window size of job j, that is, $d_j - a_j$, follows an exponential distribution, independent of other jobs. Each job has 100 units of tasks and a parallelism bound of 20. The utility of job j is modeled as a concave function of the form $f_j(x_j) = v_j x_j^\alpha$ with $\alpha < 1$. Here, we consider $\alpha = 1/2$ and the coefficient v_j is generated from a uniform distribution in [1,5]. We have also tried other types of concave functions including the logarithmic function and observed similar results. Each job is either CPU-heavy or memory-heavy with equal chance. Each task of a CPU-heavy job requires R_1 units of CPU and 1 unit of memory, and each task of a memory-heavy job requires 1 unit of CPU and R_2 units of memory, where R_1 and R_2 are uniformly distributed in $[1, R]$. Each figure below shows the average results over 10 independent scenarios generated for a given set of parameters.

Continuous Tasks: In the continuous setting, we compare Algorithm 1 with the offline optimal and two heuristics. The first heuristic, called s-greedy, always serves the jobs with highest marginal values (subject to the parallelism constraint) and ignores the resource requirements of jobs. The second heuristic, called equal-opp, schedules all the jobs with equal opportunity (subject to the parallelism constraint). Figure 2(a) shows the utility gains of these algorithms under different job availability window sizes. We observe that all the algorithms achieve higher utility gains when the availability window size increases, that is when jobs become less delay constrained, which is expected. Algorithm 1 always performs much better than the two heuristics. Moreover, it is close to the offline optimal for smaller availability windows. The gap becomes bigger for large window size due to the deadline oblivious nature of the algorithm.

Discrete Tasks: In the discrete setting, we consider two versions of Algorithm 2 that solve the local problem using the local greedy algorithm (m-greedy for short), and the primal-dual approach (primal-dual for short), respectively. We compare them with the discrete counterparts of the two heuristics in the continuous setting mentioned above. In Figure 2(b), we again vary the availability window size and observe similar result as in the continuous case. In Figure 2(c), we vary the value of R that measures the heterogeneity of resource demands. We observe that our algorithms achieve bigger improvement when R becomes larger, which indicates the benefit of being resource aware. We further observe that the two versions of Algorithm 2 have similar performance, which is consistent with our discussion in Section IV-B for small m.

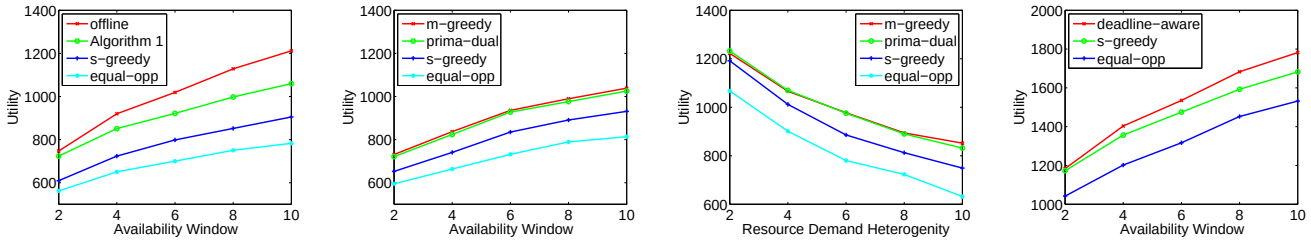
We further compare Algorithm 3 (deadline-aware for short) with the two heuristics in the single resource unit demand setting. We observe that our algorithm achieves a bigger improvement over the greedy algorithm for a larger availability window due to its deadline awareness.

VI. Related Work

Earlier works on deadline-sensitive job scheduling with partial execution focus on linear utility functions [10], [11], [13]. More recently, partial execution with non-linear utility functions has been considered in [18], [32]. However, all these works consider the CPU resource only, and none of them consider the parallelism bound. In addition, all these works apply a charging argument in establishing their competitive ratio results, which cannot be easily extended to the more complicated setting that we consider.

Deadline-sensitive job scheduling under the full execution mode has also been considered [19], [23], [25]. In this setting, constant approximation ratios are only known for some special cases. Recently, an interesting online algorithm is proposed in [25] with its competitive ratio parameterized by the slackness of jobs, where a pending job can preempt a running job only if its value is significantly higher. We expect that this idea can potentially be combined with our approaches to address more complicated utility functions such as sigmoid utilities [30].

Multi-resource sharing has received a lot of attention recently since the seminal work [15]. Previous works on multi-resource sharing mainly focus on the offline case [20], [26]. The main objective is to design new fairness metrics that are suitable to the multi-resource setting. Note that a concave utility function can also be used to model fairness in addition



(a) Continuous tasks with varying availability window size

(b) Discrete tasks with varying availability window size

(c) Discrete tasks with varying resource demand heterogeneity

(d) Discrete tasks with single resource and unit demand

Fig. 2: Simulation Results. In (a) and (b), $R = 8$. In (c), the mean availability window size is 6.

to efficiency. In particular, the multi-resource version of α-fairness proposed in [20] is a concave function. An online multi-resource sharing problem is considered in [21], where it is assumed that once an agent arrives, it will never leave the system, and there is no deadline constraint.

VII. Conclusion

Partial execution is emerging as a promising approach to improve the efficiency of cloud systems by exploiting the fact that many applications are deadline sensitive, and prefer a timely result with a good match. In this paper, we study the problem of online multi-resource allocation for serving deadline sensitive jobs with partial values. Previous works on partial execution mainly focus on CPU resource only. We consider the problem in the context of multi-resource sharing, an important paradigm for achieving fine grained resource sharing in cloud systems. Assuming that the utility functions are concave and the jobs are preemptive, we have designed efficient online algorithms with low competitive ratios.

References

[1] G. Ananthanarayanan, M. C.-C. Hung, X. Ren, I. Stoica, A. Wierman, and M. Yu. Grass: Trimming stragglers in approximation analytics. In *Proc. of NSDI*, 2014.

[2] D. P. Bertsekas. *Nonlinear Programming, 2nd edition*. Athena Scientific, 1999.

[3] A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 2005.

[4] K. M. Bretthauer and B. Shetty. The nonlinear knapsack problem algorithms and applications. *European Journal of Operational Research*, 138:459–472, 2002.

[5] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. *SIAM Journal on Computing*, 40(6):1587–1622, 2011.

[6] J. Brutlag. Speed matters for Google web search. http://services.google.com/fh/files/blogs/google_delayexp.pdf, 2009.

[7] N. Buchbinder and J. Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2007.

[8] E.-C. Chang and C. Yap. Competitive online scheduling with level of service. *Journal on Scheduling*, 6:251–267, 2003.

[9] C. Chekuri and S. Khanna. On multi-dimensional packing problems. *SIAM Journal on Computing*, 33(4):837–851, 2004.

[10] F. Y. Chin, M. Chrobak, S. P. Fung, W. Jawor, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *Journal on Scheduling*, 4:255–276, 2006.

[11] F. Y. L. Chin and S. P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37:149–164, 2003.

[12] M. Chrobak, L. Epstein, J. Noga, J. Sgall, R. van Stee, T. Tichý, and N. Vakhania. Preemptive scheduling in overloaded systems. In *Proc. of ICALP*, 2002.

[13] M. Chrobak, L. Epstein, J. Noga, J. Sgall, R. van Stee, T. Tichý, and N. Vakhania. Preemptive scheduling in overloaded systems. *Journal of Computer and System Sciences*, 67:183–197, 2003.

[14] N. R. Devanur and Z. Huang. Primal dual gives almost optimal energy efficient online algorithms. In *Proc. of SODA*, 2014.

[15] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and S. Shenker. Dominant resource fairness: Fair allocation of multiple resource types. In *Proc. of NSDI*, 2011.

[16] A. Gupta, R. Krishnaswamy, and K. Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. In *10th Workshop on Approximation and Online Algorithms (WAOA)*, 2012.

[17] B. Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Conference on Information Sciences and Systems*, 2001.

[18] Y. He, S. Elnikety, and C. Y. James Larus. Zeta: Scheduling interactive services with partial execution. In *Proc. of SoCC*, 2012.

[19] N. Jain, I. Menache, J. S. Naor, and J. Yaniv. Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters. *ACM Transactions on Parallel Computing*, 2(1), 2015.

[20] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Transactions on Networking*, 21(6):1785–1798, 2013.

[21] I. Kash, A. D. Procaccia, and N. Shah. No agent left behind: Dynamic fair division of multiple resources. *Journal of Artificial Intelligence Research*, 51:579–603, 2014.

[22] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in qos switches. In *Proc. of STOC*, 2001.

[23] G. Koren and D. Shasha. D^{over}: an optimal on-line scheduling algorithm for overloaded real-time systems. In *Real-Time Systems Symposium*, 1992.

[24] G. Linden. Marissa Mayer at Web 2.0. http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html, 2006.

[25] B. Lucier, I. Menache, J. S. Naor, and J. Yaniv. Efficient online scheduling for deadline-sensitive jobs. In *Proc. of SPAA*, 2013.

[26] D. C. Parkes, A. D. Procaccia, and N. Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. *ACM Transactions on Economics and Computation*, 3(1):1785–1798, 2015.

[27] E. Schurman and J. Brutlag. The user and business impact of server delays, additional bytes, and http chunking in web search. In *OReilly Velocity Web Performance and Operations Conference*, 2009.

[28] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau. An online auction framework for dynamic resource provisioning in cloud computing. In *Proc. of Sigmetrics*, 2014.

[29] S. Souders. Velocity and the bottom line. http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html, 2009.

[30] V. Srivastava and F. Bullo. Knapsack problems with sigmoid utilities: Approximation algorithms via hybrid optimization. *European Journal on Operational Research*, 236(2):488–498, 2014.

[31] L. Zhang, Z. Li, and C. Wu. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *Proc. of INFOCOM*, 2014.

[32] Y. Zheng, B. Ji, N. B. Shroff, and P. Sinha. Forget the deadline: Scheduling interactive applications in data centers. In *Proc. of IEEE Cloud*, 2015.

[33] Z. Zheng and N. B. Shroff. Online welfare maximization for electric vehicle charging with electricity cost. In *Proc. of ACM e-Energy*, 2014.