Scheduling Frameworks for Cloud Container Services

Ruiting Zhou, Zongpeng Li, Chuan Wu

***Abstract*—Compared to traditional virtual machines, cloud containers are more flexible and lightweight, emerging as the new norm of cloud resource provisioning. We exploit this new algorithm design space, and propose scheduling frameworks for cloud container services. Our offline and online schedulers permit partial execution, and allow a job to specify its job deadline, desired cloud containers, and inter-container dependence relations. We leverage the following classic and new techniques in our scheduling algorithm design. First, we apply the compact-exponential technique to express and handle nonconventional scheduling constraints. Second, we adopt the primal-dual framework that determines the primal solution based on its dual constraints in both the offline and online algorithms. The offline scheduling algorithm includes a new separation oracle to separate violated dual constraints, and works in concert with the randomized rounding technique to provide a near-optimal solution. The online scheduling algorithm leverages the online primal-dual framework with a learning based scheme for obtaining dual solutions. Both theoretical analysis and trace-driven simulations validate that our scheduling frameworks are computationally efficient and achieve close-to-optimal aggregate job valuation.**

***Index Terms*—Cloud Computing; Scheduling; Compact Exponential Optimization; Approximation Algorithms.**

I. INTRODUCTION

Cloud computing provides shared computing resources on demand with minimum management overhead. Cloud resources, including CPU, RAM, disk storage and bandwidth, used to be packed into different types of virtual machines (VMs) to serve different computing jobs. Launching a VM instance requires running of a full, dedicated operating system, which often consumes extra resources and takes minutes or even longer [31]. More recently, cloud containers offer an alternative to VMs. Containers are more flexible and lightweight, promising a streamlined, easy-to-deploy method of resource management. Relying on encapsulated applications, container service requires no dedicated operating system. A cloud container is able to operate with the minimum amount of resources and start in microseconds [32]. Container services available on the cloud market today include Google Container Engine [15], Amazon EC2 Container service (ECS) [5], Aliyun Container Service [4], and Azure Container Service [23].

R. Zhou is with SKLSE, School of Computer Science, Wuhan University, Wuhan, China (e-mail: rzho@ucalgary.ca).

Z. Li is with SKLSE, School of Computer Science, Wuhan University, Wuhan, China (e-mail: zongpeng@ucalgary.ca).

C. Wu is with the University of Hong Kong, Kowloon, Hong Kong (e-mail: cwu@cs.hku.hk).

This project was supported in part by NSFC (61628209, 61571335), by Hubei Science Foundation (2016CFA030, Major Project CXZD2017000121), and by the Research Grants Council of Hong Kong (17204715, 17225516, C7036-15G).

A complex computing job consists of multiple subtasks, each requiring a different configuration of cloud resources. A customized cloud container can be created accordingly to serve each subtask based on a user-defined resource profile. A subtask may depend on another, and can start execution only after the latter is completed. Such dependencies can be captured by a *dependence* graph. For example, a service chain in Network Function Virtualization (NFV) is composed of a sequential chain of virtualized network functions (VNFs) [16]. An image rendering job creates a 2D raster representation of a 3D model. As shown in its dependence graph in Fig. 1, it is composed of four subtasks to be executed sequentially: vertex processing, clipping and primitively assembling, rasterizing and fragment processing [30]. Tailor-made cloud services are available to such jobs. For instance, Azure Batch [24] is a service from Microsoft Azure, for batch processing in the cloud. A user first creates a batch job in its account and then initializes the job, including creating subtasks, configuring the container for each subtask, defining schedules and dependencies of subtasks.

While some computing jobs are time-sensitive, requiring full execution before the deadline, other jobs are elastic, and can be partially executed to obtain partial values. For example, a partially completed web searching job may return the top search results in a short time period, which is often good enough for the users [35]. After finishing the first subtask in an image rendering job, the shape of the 3D model has been outlined by vertices [30], which already provides useful information to the user. The new model of partial value for partial execution is first described as a Quality-of-Service (QoS) problem concerning the visualization of large images across a network [10]. It has applications in numerical computation, heuristic search, and database query processing [11]. Scheduling of computing jobs with partial values in the cloud has attracted recent attention from the literature [22], [7], [36], [35].

We extend the existing literature in cloud resource provisioning, and propose the first offline and online scheduling frameworks for cloud container services. We simultaneously target the following goals. **First**, we require the schedulers to be time efficient, running in polynomial time. **Second**, the aggregate value of jobs that are completed before their deadlines should be maximized. **Third**, the schedulers permit partial execution and can handle general type of jobs, *i.e.,* jobs with multiple subtasks, defined by i) the dependence graph that captures the dependence of subtasks; ii) the resource profile of each container, which is dedicated to each subtask; iii) the deadline for job completion; iv) the value of each subtask.

We formulate the offline optimization problem into a natural

Integer Linear Program (ILP). While polynomial in size, this ILP involves non-conventional scheduling constraints that are hard to be handled by the classic primal-dual framework. We apply the *compact-exponential* technique [37] to reformulate the problem into a *compact-exponential ILP*, which is a conventional packing-type ILP with an exponential number of variables corresponding to valid schedules. This compact-exponential ILP and its dual form the foundation of our offline and online scheduling algorithm design. We will show that the substantially amplified ILP size can be managed through the primal-dual technique, for computing a close-to-optimal aggregate job valuation in polynomial time.

We first assume job information is known in advance and focus on the offline scheduling algorithm design under resource capacity and job scheduling constraints. Besides serving as a benchmark for our online algorithm, the offline algorithm is also applicable to a limited near-future time window for which job information can be predicted. We leverage the classic randomized rounding technique [26]. Given a fractional solution to the LP relaxation of the compact-exponential ILP, we round the fractional solution to an integer solution by interpreting the fractional values as probabilities of schedules. The obstacle is that the compact-exponential LP relaxation is exponential in size. We resort to its dual that has a polynomial number of variables and an exponential number of constraints. We then employ the ellipsoid algorithm [9] and design a new separation oracle to separate violated constraints. The primal variables corresponding to the violated dual constraints can be selected. Consequently, we derive a new polynomial-sized LP from the original compact-exponential LP, which can be solved in polynomial time. We show that the obtained integer solution guarantees an expected $(1-\epsilon_1)$-optimal objective value, where ϵ_1 can be arbitrarily close to 0.

We proceed to consider the practical online scheduling version of the problem with stochastic input, and determine the schedule upon the arrival of each job without future information. We apply the primal-dual framework of algorithm design for such online decision making, with dual variables indicating resource prices. To address the experiential size of the compact-exponential LP, we first convert the optimization problem in the online stochastic model into a deterministic fractional program, exploiting the job arrival process. This new program removes the time domain and has a polynomial number of variables. It serves as an upper-bound of the optimal objective value in expectation, and its dual variables act as threshold of job admission. To approximately obtain a dual solution close to the offline dual optimum, we gradually learn it based on past jobs, and refine it as more jobs arrive. Our online scheduling framework guarantees computational efficiency, and produces a near-optimal expected objective value with a competitive ratio of $(1 - O(\epsilon_2))$, where ϵ_2 can be arbitrarily close to 0.

In the rest of the paper, we discuss related work in Sec. II. We introduce the system model and formulate the optimization problem in Sec. III. Sec. IV and Sec. V present the offline and online scheduling frameworks, respectively, which are evaluated in Sec. VI. Sec. VII concludes the paper.

II. RELATED WORK

Recent literature on cloud computing witnessed a plethora of studies on dynamic VM provisioning, in both offline and online settings [33][27][34]. Zhang *et al.* [33] apply a convex decomposition technique to design a randomized algorithm for dynamic cloud resource provisioning, achieving a small approximation ratio. Shi *et al.* [27] further extend the study to an online scenario, where each cloud user is subject to its budget constraint. Zhang *et al.* [27] propose an online algorithm for the stochastic job arrival model. They aim to optimize the packing of VMs to satisfy each job's demand in a fixed time window. The above studies do not consider the scheduling dimension in their solution space. Furthermore, they focus on the allocation of VMs, while we describe a richer model where each job runs over containers confined by a dependence graph.

Towards job scheduling under the full execution mode, Baruah *et al.* [8] study the traditional all-or-no-value model, and prove a tight bound on the competitive ratio for the online scheduling problem. Koren *et al.* [21] propose D-over, an algorithm that achieves the same competitive ratio. The above literature consider only one type of resource. Zhou *et al.* [37] develop online scheduling algorithms for cloud computing jobs with soft deadlines. Their design relies on information of the maximum and the minimum unit value of resources, which can be hard to obtain in the online setting.

Earlier studies on partial job execution mode often assume no resource sharing and focus on preemptive scheduling [13] [12]. Recent studies start to investigate cloud jobs with partial values. Navendu *et al.* [19] design two scheduling mechanisms for computing jobs with deadlines in the offline scenario. They consider only one type of resource, and guarantee an approximation ratio that is relatively weak. Lucier *et al.* [22] propose online scheduling algorithms for deadline-sensitive jobs in a simple model, where each job contains a single subtask. Azar *et al.* [7] further improve the algorithm and analyze its competitive ratio. Both studies assume that one server can only execute one job at each time slot. Zhang *et al.* [36] design online multi-resource allocation algorithms that allow partial execution of jobs and achieve low competitive ratios. Their model assume that all subtasks of a job are identical and have no inter-dependence. This work aims to design general scheduling frameworks for cloud container services, targeting small approximation ratio and competitive ratio in the offline and online settings, respectively.

Our offline algorithm combines the ellipsoid algorithm [9] with the randomized rounding technique [26], which is partially inspired by Fleischer *et al.*'s work [14]. However, they focus on rather different problems – maximum general assignment problems. For theoretical research on online stochastic algorithm design, Agrawal *et al.* [2] study a general online packing problem, and propose a simpler and fast primal-dual algorithm for it. They reply on a one-time learning process while our work performs a dynamic learning process. Kesselheim *et al.* [20] study online packing LPs in the random order model. They solve an LP in every step and round the fractional solution to an integer solution. Gupta *et al.* [17] consider the

problem of solving packing/covering LPs online, and construct primal solutions based on dual solutions through a regret-minimizing online learning algorithm. Jaillet *et al.* [18] study the online dynamic resource allocation problem, and propose a learning-base algorithm. Agrawal *et al.* [3] apply a similar idea on the general online optimization problem. Different from above literature [2][20][17][18][3], we do not require the number of inputs to be known in advance. Furthermore, prior work considers a more general form of the problem but limits the number of schedules for each job to a small number. Those techniques can suffer from exponential blowup in problem size when considering jobs with subtasks, as each job has an exponential number of possible schedules. In this work, we focus on a particular form of packing problem that formulates the scheduling problem for cloud container services, develop methods that are more computationally tractable and better tailored to those settings, and then evaluate those methods empirically.

III. SYSTEM MODEL

We consider a cloud service provider, which hosts a pool of K types of resources, as exemplified by CPU, RAM and disk storage. Cloud resources can be dynamically packed to different containers on demand. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. For each type-k ($k \in [K]$) resource, there is a total of c_k units in the cloud.

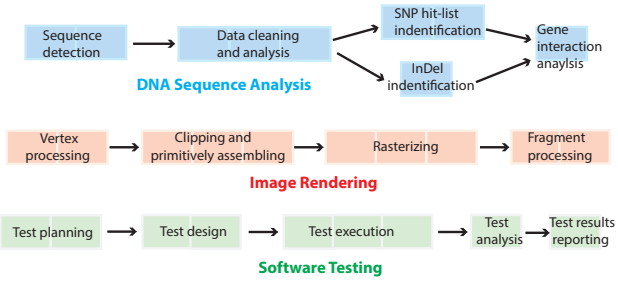


Fig. 1. Dependence graphs for cloud computing jobs.

Assume the job arrival process during a large time span $[T] = 1, 2, \ldots, T$ is a Poisson process with rate λ. Recall that a Poisson process has the following properties [29]: i) the total number of job arrivals in T time slots, I, is a random variable following Poisson distribution with an expected value of λT; ii) the arrival time of each job can be uniformly and independently mapped to a slot in $[T]$. Note that our online algorithm design relies on this assumption. However, we don't require that the job arrival process must follow a Poisson process. Our online algorithm can work on more general arrival processes, as long as the expectation of I can be estimated and property ii) holds. Based on ii), we assume the arrival time of each job is uniformly and independently drawn from $[T]$, and index jobs according to their order of arrival in any fixed realization of the arrival process. Let $[I] = \{1, 2, \ldots, I\}$ be the set of jobs. Each job i consists of multiple subtasks, and is expressed by a tuple

$$\Gamma_i = \{a_i, d_i, N_i, G_i, \{L_{in}\}_{n\in[N_i]}, \{R_{in}\}_{n\in[N_i]}, \{b_{in}\}_{n\in[N_i]}\},$$

where a_i and d_i is the arrival time and the deadline of job i. N_i is the number of subtasks in job i. G_i is the dependence graph that captures the dependencies among subtasks in job i. Example dependence graphs are illustrated in Fig 1. The execution of job i's nth subtask doesn't need to be continuous; we only require that the total execution time accumulates to L_{in}. $R_{in} = \{r^k_{in}\}_{k\in[K]}$ is the resource profile of the container that serves job i's nth subtask, where r^k_{in} is the amount of type-k resource required. If job i's nth subtask is completed by d_i, a *partial value* b_{in} is obtained. Let $r^k_{max} = \max_{i\in[I],n\in[N_i]}\{r^k_{in}\}$ denote the maximum type-k resource demand. We refer to $C = \min_{k\in[K]}\{\frac{c_k}{r^k_{max}}\}$ as *the capacity ratio*. Let $N = \max_{i\in[I]}\{N_i\}$, $D = \max_{i\in[I]}\{d_i - a_i\}$ and $L_{\max} = \max_{i\in[I],n\in[N_i]}\{L_{in}\}$. Table I summarizes notation for easy reference. Each job $i \in [I]$ is drawn independently from a set of job types, $\mathcal{D}$, following an unknown distribution, *i.e.,* job types are *i.i.d.* A job type defines the configuration of a job, including the profiles of its subtasks, *i.e.,* $\{N_i, G_i, \{L_{in}\}_{n\in[N_i]}, \{R_{in}\}_{n\in[N_i]}, \{b_{in}\}_{n\in[N_i]}\}$, and the duration of the job, *i.e,.* $d_i - a_i$. Note that a job's arrival time a_i and deadline d_i are not part of the job type. For example, an access service chain job is configured by "Firewall→IDS→Proxy" with $d_i - a_i = 20$, where instances of firewall, IDS and proxy are encapsulated into containers with predefined resource demands, and it must be deployed within 20 time slots following its arrival.

In practice, there are jobs that render an atomic value B_i only upon completion of all its subtasks before the deadline. This type of jobs can be viewed as a special case of our model, by setting $b_{i1} = b_{i2} = \cdots = b_{iN_i-1} = 0$ and $b_{iN_i} = B_i$.

Our objective is to maximize the total valuation obtained from all jobs, subjected to resource capacity and job scheduling constraints. A binary number $x_{in} \in \{0, 1\}$ indicates whether job i's nth subtask is completed (1) or not (0). Let another binary number $y_{in}(t)$ encode the scheduling of job i's nth subtask, where $y_{in}(t) = 1$ if job i's nth subtask is executed at time slot t and 0 otherwise. Under a fixed realization of the job arrival process, the offline optimization problem can be formulated into the following integer linear program (ILP):

$$\text{maximize} \quad \sum_{i\in[I]}\sum_{n\in[N_i]} b_{in}x_{in} \tag{1}$$

subject to:

$$\sum_{t=a_i}^{d_i} y_{in}(t) \geq L_{in}x_{in}, \forall i \in [I], \forall n \in [N_i], \tag{1a}$$

$$ty_{in}(t) < t'y_{in'}(t'), \forall t : y_{in}(t) = 1, \forall t' : y_{in'}(t') = 1, \forall i : n \text{ is } n''\text{s ancestor}, \tag{1b}$$

$$\sum_{i\in[I]}\sum_{n\in[N_i]} r^k_{in}y_{in}(t) \leq c_k, \forall k \in [K], \forall t \in [T], \tag{1c}$$

$$x_{in}, y_{in}(t) \in \{0, 1\}, \forall i \in [I], \forall n \in [N_i], \forall t \in [a_i, d_i]. \tag{1d}$$

Constraints (1a) guarantee that the number of allocated time slots between job i's arrival time and deadline is sufficient to serve its nth subtask. Constraints (1b) enforce the execution sequence of job i's subtasks based on its dependence graph. The capacity of type-k resource is formulated in constraints (1c).

Even in the offline setting, with complete knowledge of the system given, the polynomial-sized ILP (1) without constraints (1a) and (1b) is still a NP-hard problem, which degrades to the classic knapsack problem known to be NP-hard. The challenge further escalates when we involve unconventional job scheduling constraints (constraints (1a) and (1b). To address these challenges, we first apply the *compact-exponential* technique [37] to reformulate ILP (2) into an equivalent conventional

ILP with packing structure, at the price of introducing an exponential number of variables:

$$P: \quad \text{maximize} \quad \sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}x_{il} \tag{2}$$

subject to:

$$\sum_{i\in[I]}\sum_{l:t\in T(l)} f_{il}^k(t)x_{il} \le c_k,\ \forall k\in[K], \forall t\in[T], \tag{2a}$$

$$\sum_{l\in\zeta_i} x_{il} \le 1,\ \forall i\in[I], \tag{2b}$$

$$x_{il}\in\{0,1\},\ \forall i\in[I], \forall l\in\zeta_i. \tag{2c}$$

In the above *compact-exponential* ILP, ζ_i is the set of feasible time schedules for job i. A feasible time schedule is a vector $l=\{y_{in}(t)\}$ that satisfies constraints (1a) and (1b). b_{il} is the value based on the number of completed subtasks. $T(l)$ records the set of time slots in l. $f_{il}^k(t)$ denotes the total type-k resource occupation of job i's schedule l in t. Constraints (2a) are equivalent to (1c). Constraints (2b) ensure that each job is executed according to at most one schedule.

We relax $x_{il}\in\{0,1\}$ to $x_{il}\ge 0$, and introduce dual variables $p_k(t)$ and u_i to constraints (2a) and (2b). The dual of the relaxed problem (2) is:

$$D: \quad \text{minimize} \quad \sum_{t\in[T]}\sum_{k\in[K]} c_k p_k(t) + \sum_{i\in[I]} u_i \tag{3}$$

subject to:

$$u_i \ge b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f_{il}^k(t)p_k(t),\ \forall i\in[I], \forall l\in\zeta_i, \tag{3a}$$

$$p_k(t), u_i \ge 0, \forall i\in[I],\ \forall k\in[K], \forall t\in[T]. \tag{3b}$$

It is clear that a feasible solution to ILP (2) has a corresponding feasible solution in ILP (1), and the two ILPs have the same optimal objective value. Our offline algorithm design doesn't reply on any assumption on the job arrival process and job types, while our online algorithm design resorts to the help of them and considers the expected version of the original problem. We first focus on the offline scenario where all jobs are known in advance.

IV. Offline Scheduling Framework

In this section, we design a randomized scheduling algorithm for the offline setting, when future job information are available or can be predicted. We first solve the LP relaxation of compact-exponential ILP (2) approximately in Sec. IV-A, and then round a fractional solution of it to a feasible integer solution of ILP (1) in Sec. IV-B.

A. Solving the Compact-exponential LP

ILP (2) has an exponential number of variables, each corresponding to a possible schedule for job i. To solve ILP (2), we first solve its dual problem (3), which has a polynomial number of variables but an exponential number of constraints. We rewrite LP (3) to the following covering problem:

$$\text{minimize} \quad \sum_{t\in[T]}\sum_{k\in[K]} c_k p_k(t) + \sum_{i\in[I]} u_i \tag{4}$$

$$\text{subject to:} \quad (u_i, \{p_k(t)\}_{k\in[K],t\in[T]}) \in P_i,\ \forall i\in[I], \tag{4a}$$

$$p_k(t), u_i \ge 0, \forall i\in[I], \forall k\in[K],\ \forall t\in[T]. \tag{4b}$$

TABLE I
Summary of Notations

I	# of jobs	$[X]$	integer set $\{1,\dots,X\}$
T	# of time slots	K	# of types of resources
λ	job arrival rate	$\mathcal{D}$	job types set
a_i	job i's arrival time	d_i	job i's deadline
C	$\min_{k\in[K]}\{\frac{c_k}{r_{max}^k}\}$	r_{max}^k	$\max_{i\in[I],n\in[N_i]}\{r_{in}^k\}$
N	$\max_{i\in[I]}\{N_i\}$	D	$\max_{i\in[I]}\{d_i-a_i\}$
S	$\log_2(\frac{1}{\epsilon_2})-1$	$L_{\max}$	$\max_{i\in[I],n\in[N_i]}\{L_{in}\}$

c_k	capacity of type-k resource
N_i	# number of subtasks/containers of job i
G_i	job i's dependence graph
L_{in}	# of time slots requested by job i's nth subtask
r_{in}^k	demand of type-k resource by job i's nth subtask
b_{in}	value of job i's nth subtask
x_{in}	job i's nth subtask is completed (1) or not (0)
$y_{in}(t)$	whether or not to allocate job i's nth subtask in t
$f_{il}^k(t)$	type-k resource occupation of job i's schedule l in t

Where P_i is the polytope for job i defined by constraints of the form $u_i \ge b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f_{il}^k(t)p_k(t)$ for all $\forall l\in\zeta_i$. We resort to a separation oracle for P_i, *i.e.,* an algorithm that, given an input of dual variables $(u_i, \{p_k(t)\}_{k\in[K],t\in[T]})$, returns either a violated constraint, or guarantees that $(u_i, \{p_k(t)\}_{k\in[K],t\in[T]})$ is feasible for P_i.

If we interpret $p_k(t)$ as the marginal price of type-k resource at time t, then $b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f_{il}^k(t)p_k(t)$ is the utility of job i executed by schedule l. We can use a scheduling algorithm for the utility maximization problem for job i to design a separation oracle for P_i, as follows. Given the marginal price $\{p_k(t)\}_{k\in[K],t\in[T]}$, utility maximization for job i requires finding a schedule l^* with value u_i^* such that for any schedule $l\in\zeta_i$, $u_i^* = b_{il^*} - \sum_{k\in[K]}\sum_{t\in T(l^*)} f_{il^*}^k(t)p_k(t) \ge b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f_{il}^k(t)p_k(t)$. Then either $u_i < u_i^*$ or $u_i \ge u_i^*$. If $u_i < u_i^*$, a violated constraint with schedule l^* is found, or otherwise, $u_i \ge u_i^* \ge b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f_{il}^k(t)p_k(t)$ for any $l\in\zeta_i$. In the later case, $(u_i, \{p_k(t)\}_{k\in[K],t\in[T]})$ is feasible for P_i.

We focus on a special type of jobs with a sequential chain structure, which are usually adopted by service chains in the recent paradigm of NFV [16]. Generalization to jobs with general directed acyclic graphs is left as future work. Algorithm 1 is a separation oracle for P_i, which exactly solves the utility maximization problem for job i. The construction of the best schedule which maximizes job i's utility is based on a dynamic programming approach. We first calculate the price of container n running at time t in line 1. Because job i consists of N_i subtasks each with a partial value b_{in}, we use a `for` loop (lines 2-20) to compute the best schedule l_η if η subtasks are completed before the deadline. For the nth subtask, we calculate the cheapest schedule $\tau_n(t_s,t_e)$ to finish it within a given time period $[t_s,t_e]$ and its corresponding price in lines 4-9. Because the $(n-1)$th subtask must be completed before nth subtask ($n>1$), we also fix the schedule of the $(n-1)$th subtask when considering nth subtask's schedule, by

choosing the cheapest schedule which completes the $(n-1)$th subtask before t_s in lines 10-16. Lines 18-19 compute the best schedule and job i's utility if η subtasks are completed. Lines 21-25 figure out job i's final utility and output the result.

Algorithm 1 A Separation Oracle for Polytope P_i - Service Chain

Input: $(u_i, \{p_k(t)\}_{k\in[K],t\in[T]}), \Gamma_i$

1: Calculate $c_n(t) = \sum_{k\in[K]} r_{in}^k p_k(t), \forall n \in [N_i], t \in [a_i, d_i]$;
2: **for** $\eta = 1, \ldots, N_i$ **do**
3: **for** $n \in [\eta]$ **do**
4: **for** $t_s \in [a_i + \sum_1^{n-1} L_{in}, d_i - \sum_n^{\eta} L_{in} + 1]$ **do**
5: **for** $t_e \in [t_s + L_{in} - 1, d_i - \sum_{n+1}^{\eta} L_{in}]$ **do**
6: Select L_{in} slots between t_s and t_e with minimum $c_n(t)$, and save them to $\tau_n(t_s, t_e)$;
7: $P_n(t_s, t_e) = \sum_{t\in\tau_n(t_s,t_e)} c_n(t)$;
8: **end for**
9: **end for**
10: **if** $n > 1$ **then**
11: **for** $t_s \in [a_i + \sum_1^{n-1} L_{in}, d_i - \sum_n^{\eta} L_{in} + 1]$ **do**
12: $t_s^*, t_e^* = \arg\min_{t'_e < t_s}\{P_{n-1}(:, t'_e)\}$;
13: $P_n(t_s, t_e) = P_n(t_s, t_e) + P_{n-1}(t_s^*, t_e^*), \forall t_e$;
14: $\tau_n(t_s, t_e) = \tau_{n-1}(t_s^*, t_e^*) \cup \tau_n(t_s, t_e), \forall t_e$;
15: **end for**
16: **end if**
17: **end for**
18: $t_s^\eta, t_e^\eta = \arg\min_{t_s,t_e}\{P_\eta(t_s, t_e)\}$;
19: $l_\eta = \tau_\eta(t_s^\eta, t_e^\eta)$; $U_\eta = \sum_{n=1}^{\eta} b_{in} - P_\eta(t_s^\eta, t_e^\eta)$;
20: **end for**
21: $\eta^* = \arg\max_\eta\{U_\eta\}, l^* = l_{\eta^*}$;
22: **if** $U_{\eta^*} \geq 0$ **then**
23: **Output:** $(u_i, \{p_k(t)\}_{k\in[K],t\in[T]}) \in P_i$;
24: **else** **Output:** A violated constraint with l^*.
25: **end if**

Lemma 1: The time complexity of the separation oracle in Algorithm 1 is polynomial.

Proof: Please refer to Appendix A. □

Lemma 2: For any $\alpha > 0$, given a polynomial-time separation oracle for P_i, we can design an $(1-\alpha)$-approximation algorithm to solve the LP (3) and hence the LP relaxation of ILP (2) in polynomial time.

Proof: We run the ellipsoid method on LP (3), using Algorithm 1 as a separation oracle. More precisely, we start with an estimate of the maximum objective value of LP (3), v_0 (*e.g.,* $v_0 = \sum_{i\in[I]}\sum_{n\in[N_i]} b_{in}$), and use the ellipsoid algorithm to check the feasibility of the following linear constraints:

$$\sum_{t\in[T]}\sum_{k\in[K]} c_k p_k(t) + \sum_{i\in[I]} u_i \leq v_0,$$
$$u_i \geq b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f_{il}^k(t) p_k(t), \ \forall i \in [I], \forall l \in \zeta_i,$$
$$p_k(t), u_i \geq 0, \forall i \in [I], \ \forall k \in [K], \forall t \in [T].$$

If this LP is feasible, we know that the optimal objective value of LP (3) is at most v_0. We now decrease v_0 to $v_0/2$, and check the feasibility again. If this is true, we know the optimum lies in $(0, v_0/2]$. This is essentially a binary search to find the smallest feasible objective value. Let D^* denote the optimal objective value of LP (3). Suppose $v_0 \leq h \cdot D^*$, then after $\lfloor \log_2 h \rfloor + \lceil \log_2 \frac{1}{\alpha} \rceil$ steps, we terminate at an interval $(v^* - \alpha v^*, v^*]$, with a solution $(\{u_i\}, \{p_k(t)\})$ such that $v^* = \sum_{t\in[T]}\sum_{k\in[K]} c_k p_k(t) + \sum_{i\in[I]} u_i$. Let D be the current dual objective value and $D = v^*$. Furthermore, we have $v^* - \alpha v^* \leq D^* \leq v^*$. To check the feasibility of one point, the ellipsoid method calls the separation oracle $O(I^3\mathcal{L})$ times where each job is encoded in $\mathcal{L}$ bits [9]. Thus, we obtain a solution to LP (3) after $O(I^3(\log h + \log \frac{1}{\alpha}))$ iterations of the separation oracle. Because the running time of the separation oracle in Algorithm 1 is polynomial, the overall running time to solve LP (3) is also polynomial.

In the execution of the ellipsoid algorithm to check the feasibility of $v^* - \alpha v^*$, only a polynomial number of dual constraints (3a) are involved. This set of constraints is sufficient to show the objective value of LP (3) is greater than $v^* - \alpha v^*$. To solve the LP relaxation of ILP (2), we only need to consider a polynomial number of variables corresponding to this set of dual constraints (by setting all other variables to zero). Thus, this polynomial-sized LP can be solved in polynomial time (*e.g.,* using Karmarkar's algorithm [28], its running time is $O(I^{3.5})\mathcal{L})$. Let P be the objective value of it and $P > v^* - \alpha v^*$ by LP duality. Let P^* be the optimal objective value of the relaxed LP (2). By LP duality, $\frac{P}{P^*} \geq \frac{P}{D} > \frac{v^* - \alpha v^*}{v^*} = (1-\alpha)$, we obtain an $(1-\alpha)$-approximation algorithm. The running time of this algorithm is polynomial, which is $O(I^{3.5}(\log h + \log \frac{1}{\alpha}) K N^2 D^3 L_{\max}\mathcal{L})$ with $N = \max_{i\in[I]}\{N_i\}$ and $D = \max_{i\in[I]}\{d_i - a_i\}$. □

B. A Randomized Offline Scheduling Algorithm

Given a fractional solution to ILP (2), we continue to design a near-optimal offline algorithm to schedule jobs based on the randomized rounding technique [26]. $A_{offline}$ in Algorithm 2 is our offline scheduling algorithm. We first solve the LP relaxation of ILP (2) in line 1 using the ellipsoid method introduced in the previous subsection. Then we round the fractional solution x_{il}^f to an integer solution in lines 2-5. In order to increase the feasibility of the integer solution, we choose a schedule l^* with probability $(1 - \frac{\epsilon'}{2}) x_{il^*}^f$ for job i, where $0 < \epsilon' < 1$. We will show that with high probability (see Theorem 2), our integer solution is feasible. We first bound the probability that one of constraints (2a) is violated during the rounding of the fractional solution.

Theorem 1: **Chernoff Bound [6] [26].** Let $X_1, \ldots, X_N$ be independent Poisson trials such that, for $1 \leq n \leq N$, $Pr[X_n = 1] = p_n$, where $0 \leq p_n \leq 1$. Then for $X = \sum_{n=1}^N X_n, \mu \geq E[X] = \sum_{n=1}^N p_n$ and $0 < \delta < 2e - 1$, we have $Pr[X > (1+\delta)\mu] < e^{-\mu\delta^2/4}$.

Lemma 3: In our cloud system, assume the capacity ratio $C \geq \frac{16(c+1)}{\epsilon'^2}\ln(KT)$ with $c > 0$. Let Φ denote the event that the amount of allocated type-k resource at time t exceeds c_k, then the probability that event Φ happens is at most $\frac{1}{(KT)^{c+1}}$.

Proof: Recall that C is defined as $\min_{k\in[K]}\{\frac{c_k}{r_{max}^k}\}$ and Φ is the event that constraint (2a) is violated. For given k and t,

Algorithm 2 A Randomized offline Algorithm $A_{offline}$

Input: $\{\Gamma_i\}_{i\in[I]}, \{c_k\}_{k\in[K]}$, $0 < \epsilon' < 1$

1: Solve the LP relaxation of ILP (2) using the ellipsoid method. Let the solution be $\{x^f_{il}\}_{i\in[I],l\in\zeta_i}$;
2: **for** each job i **do**
3: Choose A schedule l^* with probability $(1-\frac{\epsilon'}{2})x^f_{il^*}$;
4: Set $x_{il^*}=1$; Update the corresponding $\{x_{in}\}_{n\in[N_i]}$ and $\{y_{in}(t)\}_{n\in[N_i],t\in[T]}$ according to schedule l^*;
5: Schedule job i accord to $y_{in}(t)$;
6: **end for**

we have

$$\begin{aligned}Pr[\Phi] &= Pr[\sum_{i\in[I]}\sum_{l:t\in T(l)} f^k_{il}(t)x_{il} > c_k]\\ &\le Pr[\sum_{i\in[I]}\sum_{l:t\in T(l)} r^k_{max}x_{il} > c_k]\\ &= Pr[\sum_{i\in[I]}\sum_{l:t\in T(l)} x_{il} > \frac{c_k}{r^k_{max}}] \le Pr[X > C],\end{aligned}$$

where $X = \sum_{i\in[I]}\sum_{l:t\in T(l)} x_{il}$. Instead of constraints (2a), we consider the following LP with new constraints $\sum_{i\in[I]}\sum_{l\in\zeta_i} x_{il} \le C$:

$$\text{maximize} \quad \sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}x_{il} \tag{5}$$

$$\text{subject to:} \quad \sum_{i\in[I]}\sum_{l\in\zeta_i} x_{il} \le C,\ \forall k\in[K], \forall t\in[T], \tag{5a}$$

$$\sum_{l\in\zeta_i} x_{il} \le 1,\ \forall i\in[I], \tag{5b}$$

$$x_{il} \ge 0,\ \forall i\in[I], \forall l\in\zeta_i. \tag{5c}$$

Let $\hat{x}^f$ be the solution to LP (5) obtained by the ellipsoid method, and x' be an integer solution to LP (5) computed by the same method in lines 3-4 in Algorithm 2. Then $Pr[x'_{il}=1] = (1-\frac{\epsilon'}{2})\hat{x}^f_{il}$. Let $X'_i = \sum_{l\in\zeta_i} x'_{il}$ and $X' = \sum_{i\in[I]} X'_i$. Then by the union bound, $Pr[X'_i = 1] \le \sum_{l\in\zeta_i} Pr[x'_{il} = 1] = (1-\frac{\epsilon'}{2})\sum_{l\in\zeta_i}\hat{x}^f_{il}$. Hence, $E[X'] = \sum_{i\in[I]} Pr[X'_i = 1] \le (1-\frac{\epsilon'}{2})C$. Let $\mu = (1-\frac{\epsilon'}{2})C$ and $\delta = \frac{\frac{\epsilon'}{2}}{1-\frac{\epsilon'}{2}}$. Because $C \ge \frac{16(c+1)}{\epsilon'^2}\ln(KT)$, $\mu \ge E[X']$ and $0 < \delta < 2e-1$, the following inequality holds by applying the Chernoff bound in Theorem 1:

$$\begin{aligned}Pr[X' > C] &< \exp\left(-(1-\frac{\epsilon'}{2})C(\frac{\frac{\epsilon'}{2}}{1-\frac{\epsilon'}{2}})^2/4\right)\\ &\le \exp\left(-\frac{c+1}{1-\frac{\epsilon'}{2}}\ln(KT)\right) = (KT)^{-\frac{c+1}{1-\frac{\epsilon'}{2}}} \le \frac{1}{(KT)^{c+1}}.\end{aligned}$$

Therefore, we obtain $Pr[\Phi] \le Pr[X > C] \le Pr[X' > C] \le \frac{1}{(KT)^{c+1}}$. □

Theorem 2: If $C \ge \frac{16(c+1)}{\epsilon'^2}\ln(KT)$, with probability at least $1-\frac{1}{(KT)^c}$, $A_{offline}$ in Algorithm 2 can output a feasible solution to ILP (1) and ILP (2) in polynomial running time. The expected value returned by it is at least $(1-\epsilon_1)$-optimal, where $\epsilon_1 = \alpha + \frac{\epsilon'}{2} - \frac{\alpha\epsilon'}{2} + \frac{1}{(KT)^c} - (\alpha + \frac{\epsilon'}{2} - \frac{\alpha\epsilon'}{2})\frac{1}{(KT)^c}$.

Proof: We first examine the feasibility and the running time. Taking a union bound on K types of resources and T time slots, the probability that the integer solution generated at line 4 in Algorithm 2 is feasible is at least $1 - KT\frac{1}{(KT)^{c+1}} = 1 - \frac{1}{(KT)^c}$ by Lemma 3. By Lemma 2, line 1 in Algorithm 2 takes polynomial time to compute a fractional solution. The running time of the `for` loop in lines 2-5 is linear. Thus, the running time of Algorithm 2 is polynomial.

Let AS denote the event that $A_{offline}$ outputs a feasible solution. Let OPT^f be the optimal objective value of the relaxed problem of (2), the expected objective value returned by Algorithm 2 is:

$$\begin{aligned}&E[\sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}x_{il}] \ge E[\sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}x_{il}|AS]\\ &\ge \sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}E[x_{il}]Pr[AS] \ge \sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}(1-\frac{\epsilon'}{2})x^f_{il}\cdot(1-\frac{1}{(KT)^c})\\ &\ge (1-\frac{\epsilon'}{2})(1-\alpha)(1-\frac{1}{(KT)^c})OPT^f = (1-\epsilon_1)OPT^f.\end{aligned}$$

Because the optimal objective value of ILP (2) is at most OPT^f, we can conclude that Algorithm 2 returns a $(1-\epsilon_1)$-optimal solution in expectation with $\epsilon_1 = \alpha + \frac{\epsilon'}{2} - \frac{\alpha\epsilon'}{2} + \frac{1}{(KT)^c} - (\alpha + \frac{\epsilon'}{2} - \frac{\alpha\epsilon'}{2})\frac{1}{(KT)^c}$. □

V. Online Scheduling Framework

A practical scheduling algorithm needs to work in the online fashion, without relying on knowledge of future job arrivals. In this section, we design an online algorithm that runs as jobs arrive to the system, and processes each job immediately upon its arrival. We next introduce the primal-dual framework that guides our online algorithm design in Sec. V-A. We propose an online algorithm for jobs with chain structure in Sec. V-B and analyze its performance in Sec. V-C. Sec. V-D shows that the algorithm proposed in Sec. V-B can also handle general jobs with directed acyclic graph structures.

A. Primal and Dual Framework

Upon each job arrival, the cloud service provider needs to determine whether to serve this job, and if so, how to schedule it. This process is equivalent to choosing a feasible solution to ILP (1). To solve ILP (1), we resort to the classic primal-dual framework, and apply it to the compact-exponential ILP (2) and its dual (3). We observe that for each primal variable x_{il}, there is a dual constraint associated to it. Complementary slackness indicates the update of the primal variable is based on its dual constraint. x_{il} remains zero unless its associated dual constraint (2a) is tight. Let $\mathbf{p}^*$ denote the optimal solution of dual variables $\{p_k(t)^*\}_{\forall k\in[K],t\in[T]}$ for LP (3). Upon the arrival of the ith job, we assign dual variable u_i to the maximum of 0 and the right hand side (RHS) of (2a),

$$u_i = \max\{0, \max_{l\in\zeta_i}\{b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)} f^k_{il}(t)p_k(t)^*\}\}. \tag{6}$$

If $u_i > 0$, the cloud service provider serves job i according to the schedule that maximizes the RHS of constraint (2a); If $u_i \le 0$, the cloud service provider rejects it. The rationale is as follows: The dual variable $p_k(t)^*$ can be interpreted as the marginal price per unit of type-k resource at time t, then $\sum_{k\in[K]}\sum_{t\in T(l)} f^k_{il}(t)p_k(t)^*$ is the price to execute job i according to schedule l. RHS of (2a) can be viewed as job i's utility with schedule l. The assignment of u_i in (6) effectively maximizes job i's utility, towards achieving the maximum value obtained from all jobs.

However, the problem is that we cannot obtain the optimal dual solution $\mathbf{p}^*$ in the online setting. We only have information on past jobs. Thus, we consider the first $\epsilon_2 \in (0,1)$ fraction of jobs and hope to obtain an approximation dual solution in expectation, and progressively refine our dual solution as more jobs arrive. By adopting this idea, we next design an online algorithm, and show that it has good performance in both theoretical analysis and simulation studies.

B. An Online Algorithm with Stochastic Input

We first focus on service chain type of jobs where the dependence graph is of a sequential chain structure.

Expected offline optimization problem. The offline problem in (2) is defined under a fixed and stochastic realization of the job arrival process. Next, we consider all possible realizations of the job arrival process in expectation, and define the expected offline problem in LP (7). We refer it as the *expected offline program*. It guides our online algorithm design and the optimal objective value of it servers as an upper bound of the expected optimal objective value of the offline problem in (2) in the competitive ratio analysis.

We use j to denote a job of type-j instead of job j in LPs (7), (8) and (9). Let ρ_j be the probability that type-j job is drawn from the job types set $\mathcal{D}$. Since the expected number of jobs is λT, the expected number of type-j job appearing in the realized jobs is $\lambda T \rho_j$. Let x_{jl} be the probability of type-j job served according to schedule l, over a random realization of jobs. Then $\lambda T \rho_j \sum_{l \in \zeta_j} b_{jl} x_{jl}$ is the contribution of type-j jobs to the expected overall obtained value. Summing over all job types, the objective function of (7) represents the expected value obtained from all jobs. Note that we assume the same type of jobs has the same value of $d_i - a_i$ regardless of job arrival time, under the assumption that T is much larger the value of $d_i - a_i$. Because the probability of $d_i > T$ is very small and the overall obtained value in expectation barely changes without considering these extreme jobs.

$$\text{maximize} \quad \sum_{j \in \mathcal{D}} \lambda T \rho_j \sum_{l \in \zeta_j} b_{jl} x_{jl} \tag{7}$$

subject to:

$$\sum_{j \in \mathcal{D}} \sum_{l \in \zeta_j} \lambda T \rho_j \frac{\sum_{t \in T(l)} f_{il}^k(t)}{T} x_{jl} \le c_k, \forall k \in [K], \tag{7a}$$

$$\sum_{l \in \zeta_j} x_{jl} \le 1, \forall j \in \mathcal{D}, \tag{7b}$$

$$x_{jl} \ge 0, \forall j \in \mathcal{D}, \forall l \in \zeta_j. \tag{7c}$$

Next, we examine the constraints in LP (7). Constraints (7a) are the expected capacity constraints, which guarantee the average consumption of one type of resource at each slot is below its capacity. The rationale is as follows: If a type-j job is scheduled according to l, then it consumes total $\sum_{t \in T(l)} f_{il}^k(t)$ units of type-k resource over the entire system time (T slots). Recall that the arrival time of a job is uniformly distributed within $[T]$, then the slot $t \in T(l)$ is also uniformly distributed within $[T]$. On average over time, a type-j job served with schedule l consumes at most $\frac{\sum_{t \in T(l)} f_{il}^k(t)}{T}$ units of type-k resource at each time slot, as the probability of this job occupying any slot is $1/T$. $\sum_{j \in \mathcal{D}} \sum_{l \in \zeta_j} \lambda T \rho_j \frac{\sum_{t \in T(l)} f_{il}^k(t)}{T} x_{jl}$ is the average consumption of type-k resource at each slot contributed by all types of jobs. Note that it is a non-trivial transformation of the capacity constraints (2a) as we remove the time dimension here. Constraints (7b) ensures that one job of a specific type can only be served according to at most one schedule. Based on the above expected offline program, we are able to design an online algorithm that obtains $1 - O(\epsilon_2)$ fraction of the expected optimal value obtained from all jobs, under the assumption that each job only consumes a small fraction of the capacity of any resource.

Although it seems that the number of variables in LP (7) is still exponential, we observe that there are only N_j possible values of b_{jl} and $\sum_{t \in T(l)} f_{il}^k(t)$ for each j. This is because a type-j job contains N_j subtasks that need to be executed sequentially, and each of the subtask has its own value and the resource demand. Let $\eta \in [N_j]$ denote the ηth execution option for type-j job, $b_{j\eta} = \sum_{n \in [\eta]} b_{jn}$ and $\omega_{j\eta}^k = \sum_{n \in [\eta]} r_{in}^k L_{in}$ represent the value and the resource consumption for this option. We can rewrite LP (7) to the following LP:

$$P_\Sigma: \quad \text{maximize} \quad \sum_{j \in \mathcal{D}} \lambda T \rho_j \sum_{\eta \in [N_j]} b_{j\eta} x_{j\eta} \tag{8}$$

subject to:

$$\sum_{j \in \mathcal{D}} \sum_{\eta \in [N_j]} \lambda T \rho_j \frac{\omega_{j\eta}^k}{T} x_{j\eta} \le c_k, \forall k \in [K], \tag{8a}$$

$$\sum_{\eta \in [N_j]} x_{j\eta} \le 1, \forall j \in \mathcal{D}, \tag{8b}$$

$$x_{j\eta} \ge 0, \forall j \in \mathcal{D}, \forall \eta \in [N_j]. \tag{8c}$$

By introducing dual variables p_k and u_j to constraints (8a) and (8b), the dual problem of (8) is:

$$D_\Sigma: \quad \text{minimize} \quad \sum_{k \in [K]} c_k p_k + \sum_{j \in \mathcal{D}} \lambda T \rho_j u_j \tag{9}$$

subject to:

$$u_j \ge b_{j\eta} - \sum_{k \in [K]} \frac{\omega_{j\eta}^k}{T} p_k, \forall j \in \mathcal{D}, \forall \eta \in [N_j], \tag{9a}$$

$$p_k, u_j \ge 0, \forall k \in [K], \forall j \in \mathcal{D}. \tag{9b}$$

If we can solve the dual problem in (9) exactly to obtain the optimal dual solution $\mathbf{p}_\Sigma$, we can apply the primal-dual technique discussed in Sec. V-A to derive the primal solution for the expected offline program (8), achieving a close-to-optimal objective value. The barrier is still that we do not have complete knowledge of all job types in the online setting. Our main idea is to produce an approximate dual solution based on past jobs, and gradually refine this dual solution with the accumulation of past jobs. The intuition is that because the types of jobs are *i.i.d.*, the average resource consumption of the past jobs can approximately reflect the average resource consumption of all jobs in expectation, especially when more and more jobs are processed. More specifically, we divide the job arrival process into $\log_2(\frac{1}{\epsilon_2})$ stages, and index each stage with an integer s. Let $S = \log_2(\frac{1}{\epsilon_2}) - 1$ and then $s \in [0, 1, \ldots, S]$. For each stage, we consider the first $2^s \lfloor \epsilon_2 \lambda T \rfloor$ jobs in set $\mathcal{I}_s = [1, \ldots, 2^s \lfloor \epsilon_2 \lambda T \rfloor]$, and formulate an empirical version of (8) in P_s in (10) for these sample jobs. We replace the expectations over all jobs in the objective function and constraints (8a) with the sum over these jobs, and shrink the capacity limits accordingly by a factor of $(1 - \mathcal{F}_s) 2^s \epsilon_2$. Let $I_s = |\mathcal{I}_s| = 2^s \lfloor \epsilon_2 \lambda T \rfloor$ and $\mathcal{F}_s = \epsilon_2 \sqrt{\frac{\lambda T}{2^s \epsilon_2 \lambda T}} = \sqrt{\frac{\epsilon_2}{2^s}}$. Then $2^s \epsilon_2 \approx \frac{I_s}{\lambda T}$ is the proportion of first $2^s \lfloor \epsilon_2 \lambda T \rfloor$ jobs to all jobs, and $(1 - \mathcal{F}_s)$ handles the sampling error to make sure the overall resource consumption does not exceed the capacity. Note that $\epsilon_2 \le \mathcal{F}_s \le \sqrt{\epsilon_2}$ and we convert each job type j back to job i. The dual of the relaxed (10) is formulated in

(11).

$$P_s: \quad \text{maximize} \quad \sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta} \tag{10}$$

subject to:

$$\sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T}x_{i\eta} \le (1-\mathcal{F}_s)2^s\epsilon_2 c_k,\ \forall k\in[K], \tag{10a}$$

$$\sum_{\eta\in[N_i]} x_{i\eta} \le 1,\ \forall i\in\mathcal{I}_s, \tag{10b}$$

$$x_{i\eta}\in\{0,1\}, \forall i\in\mathcal{I}_s,\ \forall\eta\in[N_i]. \tag{10c}$$

$$D_s: \quad \text{minimize} \quad \sum_{k\in[K]}(1-\mathcal{F}_s)2^s\epsilon_2 c_k p_k + \sum_{i\in\mathcal{I}_s} u_i \tag{11}$$

subject to:

$$u_i \ge b_{i\eta} - \sum_{k\in[K]}\frac{\omega_{i\eta}^k}{T}p_k,\ \forall i\in\mathcal{I}_s, \forall\eta\in[N_i], \tag{11a}$$

$$p_k, u_i \ge 0,\ \forall k\in[K], \forall i\in\mathcal{I}_s. \tag{11b}$$

Upon the arrival of the $2^s\lfloor\epsilon_2\lambda T\rfloor$th job, we exactly solve the dual problem in (11) to obtain the optimal dual solution $\mathbf{p}_s$. The size of the dual problem (11) is polynomial, and hence it can be solved efficiently by Karmarkar's algorithm [28]. By involving more and more jobs in solving (11), we progressively learn a dual solution that is close to the optimal dual solution $\mathbf{p}_\Sigma$ of the offline dual problem in (9).

We next discuss the decision making and the scheduling process, based on the learned dual solution $\mathbf{p}_s$. Upon the arrival of each job, we let u_i be the maximal of 0 and the RHS of constraints (11a), *i.e.,*

$$u_i = \max\{0, \max_{\eta\in[N_i]}\{b_{i\eta} - \sum_{k\in[K]}\frac{\omega_{i\eta}^k}{T}p_{k,s}\}\}.$$

If $u_i \le 0$, the cloud service provider rejects this job; If $u_i > 0$, the cloud service provider accepts this job, and serves it according to the following schedule: Let $\eta_i = \arg\max_{\eta\in[N_i]}\{b_{i\eta} - \sum_{k\in[K]}\frac{\omega_{i\eta}^k}{T}p_{k,s}\}$, subtasks $1,\ldots,\eta_i$ in job i will be allocated sequentially to slots from a_i to $a_i+\sum_{n\in[\eta_i]}L_{in}-1$. Although we didn't check the resource capacity constraints (2a) here, we show that with high probability (see Lemma 7), our algorithm satisfies the capacity limit in expectation for any type of resource at any time.

A_{online} in Algorithm 3 is our online algorithm, with the scheduling algorithm A_{core} in Algorithm 4 running for each job. Lines 1-2 in A_{online} define variable I_s and initialize primal and dual variables. Lines 4-5 reject the first $\lfloor\epsilon_2\lambda T\rfloor$ jobs as price $\mathbf{p}_0$ is not ready yet. Upon the arrival of the ith job ($i \ge \lfloor\epsilon_2\lambda T\rfloor + 1$), lines 6-13 determine whether to serve this job, and if so how to schedule it. More specifically, A_{core} in line 7 is run for job $i\in[2^{s-1}\lfloor\epsilon_2\lambda T\rfloor+1, 2^s\lfloor\epsilon_2\lambda T\rfloor]$ with the input $\mathbf{p}_{s-1}$. In A_{core}, lines 1-4 determine the utility variable u_i. If $u_i > 0$, we accept job i, compute its schedule l in line 7 and update all primal variables in lines 6-14. On the arrival of $2^s\lfloor\epsilon_2\lambda T\rfloor$th job, line 15 in A_{online} solves the dual LP (11) exactly using all jobs from job 1 to job $2^s\lfloor\epsilon_2\lambda T\rfloor$. Line 16 updates p_k and s. Note that the last time we update price $\mathbf{p}_s$ is the arrival time of job $2^{\log_2(\frac{1}{\epsilon_2})-1}\lfloor\epsilon_2\lambda T\rfloor$. This process is repeated until the last job arrives. Note that our algorithm doesn't require any information about the job type distribution. Furthermore, we can use an estimated value of λ instead of the accurate one. We will show that inaccurate estimation has rather mild impact on the performance in the simulations. We next use a simple example to illustrate the process of A_{online}. Suppose the online system spans 32 time slots. Let $\lambda = 0.5$ and $\epsilon_2 = \frac{1}{4}$. We reject the first 4 jobs, and solve (11) with the input of the first 4 jobs to obtain $\mathbf{p}_0$. From job 4 to job 8, we use $\mathbf{p}_0$ as the price to make decision and solve (11) again with the input of the first 8 jobs to obtain $\mathbf{p}_1$. From job 8 to the last job, $\mathbf{p}_1$ serves as the threshold to determine the winner.

Algorithm 3 An Online Algorithm A_{online}

Input: $\{\Gamma_i\}, \{c_k\}, \epsilon_2, \lambda, T$

1: Define $I_s = 2^s\lfloor\epsilon_2\lambda T\rfloor$;
2: Initialize $s=0$; Let $x_{in}=0, y_{in}(t)=0, x_{il}=0, u_i=0, p_k=0, \forall i\in[I], \forall n\in[N_i], \forall t\in[T], \forall l\in\zeta_i, \forall k\in[K]$ by default;
3: **while** the arrival of the ith job **do**
4: **if** $i \le \lfloor\epsilon_2\lambda T\rfloor$ **then**
5: Reject job i;
6: **else**
7: $(\{x_{in}\}, \{y_{in}(t)\}) = A_{core}(\Gamma_i, \{c_k\}, \{p_k\})$;
8: **if** $\exists n\in[N_i], x_{in}=1$ **then**
9: Schedule job i according to $y_{in}(t)$;
10: **else**
11: Reject job i.
12: **end if**
13: **end if**
14: **if** $i = I_s$ and $s \le \log_2(\frac{1}{\epsilon_2}) - 1$ **then**
15: Solve the dual LP (11) exactly to obtain $\mathbf{p}_s$;
16: Let $\{p_k\} = \mathbf{p}_s$; $s = s+1$;
17: **end if**
18: **end while**

Algorithm 4 A Scheduling Algorithm A_{core}

Input: $\Gamma_i, \{c_k\}, \{p_k\}$
Output: $\{x_{in}\}, \{y_{in}(t)\}$

1: **for** $\eta = 1, 2, \ldots, N_i$ **do**
2: $u_{i\eta} = \sum_{n\in[\eta]} b_{in} - \sum_{k\in[K]}\frac{\sum_{n\in[\eta]} r_{in}^k L_{in}}{T}p_k$;
3: **end for**
4: $u_i = \max\{0, \max_{\eta\in[N_i]}\{u_{i\eta}\}\}$;
5: **if** $u_i > 0$ **then**
6: $\eta_i = \arg\max_{\eta\in[N_i]}\{u_{i\eta}\}$;
7: $l_i = \{a_i, \ldots, a_i + \sum_{n\in[\eta_i]} L_{in} - 1\}$; $x_{il_i} = 1$;
8: $x_{in} = 1, \forall n\in[\eta_i]$; $t = t_i$;
9: **for** $n = 1, \ldots, \eta_i$ **do**
10: $index = 1$;
11: **while** $index \le L_{in}$ **do**
12: $y_{in}(t) = 1; t = t+1; index = index + 1$;
13: **end while**
14: **end for**
15: **end if**
16: **Return** $\{x_{in}\}, \{y_{in}(t)\}$

C. Theoretical Analysis

i) Polynomial running time.

Theorem 3: The time complexity of A_{online} in Algorithm 3 is polynomial.

Proof: Please refer to Appendix B. □

ii) Feasibility of the original problem.

We next show that with high probability, our online algorithm A_{online} can compute a feasible solution to original problem (2). Constraints (2b) and (2c) are satisfied trivially. When summing over all $s \in [0, \dots, S]$, Lemma 7 shows that with probability at least $1 - 2\epsilon_2$, accepted jobs consume at most the maximum capacity in expectation for any type of resource at any time (*i.e.,* Constraints (2a) are satisfied).

Let $x_{i\eta}(\mathbf{p}_s)$ be the primal solution output by A_{online}, which is a function of $\mathbf{p}_s$. We have

$$x_{i\eta}(\mathbf{p}_s) = \begin{cases} 1, \text{ if } \eta = \arg\max\limits_{\eta' \in [N_i]} \{b_{i\eta'} - \sum\limits_{k\in[K]} \frac{\omega^k_{i\eta'}}{T} p_{k,s}\} \\ \quad \text{and } b_{i\eta} > \sum\limits_{k\in[K]} \frac{\omega^k_{i\eta}}{T} p_{k,s}, \\ 0, \text{ otherwise.} \end{cases} \tag{12}$$

We next define two random variables $\mathbf{X}^k_i$ and $\mathbf{Y}^k_i(t)$, which will be used in the following analysis.

$$\mathbf{X}^k_i = \sum_{\eta\in[N_i]} \frac{\omega^k_{i\eta}}{T} x_{i\eta}(\mathbf{p}_s). \tag{13}$$

$$\mathbf{Y}^k_i(t) = \begin{cases} \sum\limits_{l\in\zeta_i} f_{il}(t) x_{il}, \text{ if } t \in T(l), \\ 0, \text{ otherwise.} \end{cases} \tag{14}$$

Note that the value of x_{il} in $\mathbf{Y}^k_i(t)$ is output by A_{online} and computed according to the value of $x_{i\eta}(\mathbf{p}_s)$.

Lemma 4: The expectation of $\mathbf{Y}^k_i(t)$ on t is upper bounded by $\mathbf{X}^k_i$ when job i's arrival time t_i is uniformly disturbed in $[T]$.

Proof: Please refer to Appendix C. □

Lemma 5: Let E_1 denote the event that the total number of jobs arrived in $[T]$, I, is within the range of $[(1-\frac{\mathcal{F}_s}{2})\lambda T, (1+\frac{\mathcal{F}_s}{2})\lambda T]$, $\forall \mathcal{F}_s$, the probability of E_1 happens is at least $1-\epsilon_2$, given $\lambda T \geq \frac{4}{(\epsilon_2)^3}$.

Proof:
$$\begin{aligned} Pr[E_1] &\geq 1 - Pr[|I - \lambda T| \geq \frac{\mathcal{F}_s}{2}\lambda T] \\ &\geq 1 - Pr[|I - \lambda T| \geq \frac{\epsilon_2}{2}\lambda T]. \end{aligned}$$

The last inequity holds because $\mathcal{F}_s \geq \epsilon_2$. According to Chebyshev's inequality [26], we can obtain

$$\begin{aligned} &Pr[|I - \lambda T| \geq \frac{\epsilon_2}{2}\lambda T] = Pr[|I - E[I]| \geq \frac{\epsilon_2}{2}\lambda T] \\ &\leq \frac{Var[I]}{(\frac{\epsilon_2}{2}\lambda T)^2} = \frac{4\lambda T}{\epsilon_2^2 \lambda^2 T^2} = \frac{4}{\epsilon_2^2 \lambda T}. \end{aligned}$$

Given $\lambda T \geq \frac{4}{(\epsilon_2)^3}$, we have $\frac{4}{\epsilon_2^2 \lambda T} \leq \epsilon_2$ and therefore $Pr[E_1] \geq 1 - \epsilon_2$. □

We define a new variable $\mathcal{B}$, and let

$$\mathcal{B} = \max\left\{ \frac{12\ln\left(2(IN)^K KT\log_2(\frac{1}{\epsilon_2})/\epsilon_2\right)}{\epsilon_2^2}, \frac{4\lambda T}{\epsilon_2^2} \right\}.$$

Lemma 6: Let E_2 denote the event that $\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} \mathbf{X}^k_i \geq 2^s\epsilon_2 c_k, \forall k \in [K], s \in [0, 1, \dots, S]$. On the condition of E_1, i.e., $(1-\frac{\mathcal{F}_s}{2})\lambda T \leq I \leq (1+\frac{\mathcal{F}_s}{2})\lambda T$, the probability of E_2 happens, $Pr[E_2|E_1]$, is at most $\frac{\epsilon_2}{T}$, given $\frac{c_k}{r^k_{max}} \geq \mathcal{B}$.

Proof: Consider a fixed price $\mathbf{p}$, we say a random sample $\mathcal{I}_{s+1}\setminus\mathcal{I}_s$ is bad for this $\mathbf{p}$ if $\mathbf{p} = \mathbf{p}_s$ but $\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} \sum_{\eta\in[N_i]} \frac{\omega^k_{i\eta}}{T} x_{i\eta}(\mathbf{p}) \geq 2^s\epsilon_2 c_k$, for some k and s. We first show that the probability of bad samples is small for every fixed $\mathbf{p}$, s and k. Then we take union bound over all "distinct" prices, all s, and all k to prove with small probability, $\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} \sum_{\eta\in[N_i]} \frac{\omega^k_{i\eta}}{T} x_{i\eta}(\mathbf{p}_s) \geq 2^s\epsilon_2 c_k, \forall k, \forall s$ with price $\mathbf{p}_s$.

We first fix $\mathbf{p}$, s and k. Recall the definition of $\mathbf{X}^k_i$ in (12). Since $\mathbf{p_s}$ is the optimal solution for LP (11), they by complementary conditions, we have $\sum_{i\in\mathcal{I}_s} \mathbf{X}^k_i \leq (1-\mathcal{F}_s)2^s\epsilon_2 c_k$. We define events $A = \{\sum_{i\in\mathcal{I}_s} \mathbf{X}^k_i \leq (1-\mathcal{F}_s)2^s\epsilon_2 c_k\}$, $B = \{\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} \mathbf{X}^k_i \geq 2^s\epsilon_2 c_k\}$, Therefore, the probability of bad samples is bounded by:

$$\begin{aligned} Pr[B] &= Pr[\sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i - \sum_{i\in\mathcal{I}_s} \mathbf{X}^k_i \geq 2^s\epsilon_2 c_k] \\ &= Pr[\sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i \geq (2-\mathcal{F}_s)2^s\epsilon_2 c_k | A] \\ &\leq Pr[|\sum_{i\in\mathcal{I}_s} \mathbf{X}^k_i - \frac{I_s}{I_{s+1}} \sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i| \geq \beta] \end{aligned} \tag{15}$$

Because $\frac{I_s}{I_{s+1}} = \frac{1}{2} \geq \frac{1}{2(1+\mathcal{F}_s/2)}$ or $\frac{I_s}{I_{s+1}} = \frac{\lambda T/2}{I} \geq \frac{1}{2(1+\mathcal{F}_s/2)}$ as $I \leq (1+\frac{\mathcal{F}_s}{2})\lambda T$, thus,

$$\begin{aligned} &|\sum_{i\in\mathcal{I}_s} \mathbf{X}^k_i - \frac{I_s}{I_{s+1}} \sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i| \geq \\ &(\frac{1}{1+\mathcal{F}_s/2}(1-\mathcal{F}_s/2) - (1-\mathcal{F}_s))2^s\epsilon_2 c_k \\ &= \frac{\frac{\mathcal{F}_s^2}{2}}{1+\mathcal{F}_s/2} 2^s\epsilon_2 c_k \geq \frac{\mathcal{F}_s^2}{4} 2^s\epsilon_2 c_k. \end{aligned}$$

Then $\beta = \frac{\mathcal{F}_s^2}{4} 2^s\epsilon_2 c_k$.

We normalize r^k_{max} such that $\mathbf{X}^k_i \in [0,1]$, and replace c_k with $\frac{c_k}{r^k_{max}}$. We define random variables:

$$\sigma^2(\mathbf{X}) = \frac{1}{I_{s+1}} \sum_{i\in\mathcal{I}_{s+1}} (\mathbf{X}^k_i - \frac{1}{I_{s+1}} \sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i)^2 \leq 1.$$

$$\Delta(\mathbf{X}) = \max_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i - \min_{i\in\mathcal{I}_{s+1}} \mathbf{X}^k_i \leq 1.$$

According to Hoeffding-Berstein Inequality (Appendix A.1 in [3]), we have

$$\begin{aligned} (15) &\leq 2\exp\left(-\frac{\beta^2}{2I_s\sigma^2(\mathbf{X}) + \beta\Delta(\mathbf{X})}\right) \\ &\leq 2\exp\left(-\frac{\frac{\mathcal{F}_s^4}{16}2^{2s}\epsilon_2^2 c_k^2}{2I_s + \frac{\mathcal{F}_s^2}{4}2^s\epsilon_2 c_k}\right). \end{aligned} \tag{16}$$

$I_s \leq 2^s\epsilon_2\lambda T \leq \lambda T$. Because $c_k/r^k_{max} = c_k \geq 4\lambda T/\epsilon_2^2$, we have $I_s \leq \lambda T \leq \epsilon_2^2 c_k/4 = \frac{\mathcal{F}_s^2}{4}2^s\epsilon_2 c_k$. Thus,

$$\begin{aligned} (16) &\leq 2\exp\left(-\frac{\frac{\mathcal{F}_s^2}{4}2^s\epsilon_2 c_k}{2+1}\right) \leq 2\exp\left(-\frac{\epsilon_2^2 c_k}{12}\right) \\ &\leq \frac{\epsilon_2}{K(IN)^K T\log_2(\frac{1}{\epsilon_2})}. \end{aligned}$$

The last inequality holds because $c_k/r^k_{max} = c_k \geq \mathcal{B}$. Next, we take a union bound over all "distinct" $\mathbf{p}$. Two price

vectors $\mathbf{p_1}$ and $\mathbf{p_2}$ are distinct if and only if they result in distinct solution, *i.e.,* $x_{i\eta}(\mathbf{p_1}) \neq x_{i\eta}(\mathbf{p_2})$. By results from computational geometry [25], the total number of such distinct prices is at most $(IN)^K$. Taking union bound over all distinct prices, K types of resources and $\log_2(\frac{1}{\epsilon_2})$ stages, we get the desired result. □

Lemma 7: With probability at least $1-2\epsilon_2$, we have

$$\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} E[\mathbf{Y}_i^k(t)] \leq 2^s\epsilon_2 c_k, \forall k\in[K], t\in[T], s\in[0,1,\ldots,S],$$

given $\lambda T \geq \frac{4}{(\epsilon_2)^3}$ and $\frac{c_k}{r^k_{max}} \geq \mathcal{B}$.

Proof: We first prove that, for a fixed t, on the condition of E_1, the probability of $\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} E[\mathbf{Y}_i^k(t)] \geq 2^s\epsilon_2 c_k, \forall k, \forall s$ is small.

According to Lemma 4, the expectation of $\mathbf{Y}_i^k(t)$ on t is upper bounded by $\mathbf{X}_i^k$. Therefore,

$$\begin{aligned}
&Pr[\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} E[\mathbf{Y}_i^k(t)] \geq 2^s\epsilon_2 c_k, \forall k, \forall s|E_1]\\
&\leq Pr[\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} \mathbf{X}_i^k \geq 2^s\epsilon_2 c_k \forall k, \forall s|E_1]\\
&= Pr[E_2|E_1] \leq \frac{\epsilon_2}{T}. \text{ (Lemma 6)}
\end{aligned}$$

We take union bound over T slots and have

$$\begin{aligned}
&P[\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} E[\mathbf{Y}_i^k(t)] \leq 2^s\epsilon_2 c_k, \forall k, \forall s, \forall t]\\
&\geq P[\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} E[\mathbf{Y}_i^k(t)] \leq 2^s\epsilon_2 c_k, \forall k, \forall s, \forall t|E_1]Pr[E_1]\\
&\geq (1-T\cdot Pr[\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s} E[\mathbf{Y}_i^k(t)] \geq 2^s\epsilon_2 c_k, \forall k, \forall s|E_1])Pr[E_1]\\
&\geq (1-\epsilon_2)^2 \geq 1-2\epsilon_2. \text{ (Lemma 5)}
\end{aligned}$$

□

iii) Competitive Ratio.

Finally, we show that our algorithm A_{online} is a $1-O(\epsilon_2)$ competitive in expectation in Theorem 4.

Lemma 8: Let OPT denote the optimal objective value of the offline problem in (2). $E[OPT]$ is the expectation of OPT over all possible realizations of the job arrival process. The optimal objective value of LP (8) is at least $E[OPT]$.

Proof: We observe that the average of the optimal solutions of the offline problem in (2), computed over all possible realizations of the job arrival process, achieves the expected offline social welfare $E[OPT]$. Furthermore, it also provides a feasible solution to the expected offline problem in (8). Therefore, the optimal objective value of LP (8) must be at least $E[OPT]$. □

As $|\mathcal{I}_s| \leq 2^S\lfloor\epsilon_2\lambda T\rfloor \leq \frac{\lambda T}{2} < \lambda T$, we have the following observation.

Observation 1. *The inputs of the problem in (10) have the following property: for the optimal dual solution $\mathbf{p}_s$ derived by solving the dual problem (11), there can be at most λT equations such that $b_{i\eta_i} = \sum_{k\in[K]} \frac{\omega^k_{i\eta_i}}{k} p_{k,s}, \forall i \in \mathcal{I}_s$, where η_i denote the best option for job i.*

Lemma 9: Let $\{x_{i\eta,s}\}_{i\in[I],\eta\in[N_i]}$ be the optimal solution of (10), and $\mathbf{x}_s$ be the solution vector. $\sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} x_{i\eta,s} - \lambda T \leq \sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} x_{i\eta}(\mathbf{p}_s) \leq \sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} x_{i\eta,s}, \forall s \in [0,1,\ldots,S]$.

Proof: Let η_i denote the best option for job i, *i.e.,* $\eta_i = \arg\max_{\eta'\in[N_i]}\{b_{i\eta'} - \sum_{k\in[K]} \frac{\omega^k_{i\eta'}}{T} p_{k,s}\}$. By complementary slackness, the optimal solution of (10) satisfies $x_{i\eta_i,s} = 0$ if $b_{i\eta_i} < \sum_{k\in[K]} \frac{\omega^k_{i\eta_i}}{T} p_{k,s}$, and $x_{i\eta_i,s} > 0$ if $b_{i\eta_i} = \sum_{k\in[K]} \frac{\omega^k_{i\eta_i}}{T} p_{k,s}$. Compared with (12), the only difference is $x_{i\eta_i}(\mathbf{p}_s) = 0$ when $b_{i\eta_i} = \sum_{k\in[K]} \frac{\omega^k_{i\eta_i}}{T} p_{k,s}$. These imply that jobs accepted by A_{online} are also accepted by the optimal solution, while some jobs rejected by A_{online} are accepted by the optimal solution. Since Observation 1 indicates that there are at most λT equation satisfy $b_{i\eta_i} = \sum_{k\in[K]} \frac{\omega_{i\eta_i}}{k} p_{k,s}, \forall i \in \mathcal{I}_s$, there are at most λT jobs that are rejected by A_{online} but accepted by the optimal solution. □

Lemma 10: On the condition of $(1-\frac{\mathcal{F}_s}{2})\lambda T \leq I \leq (1+\frac{\mathcal{F}_s}{2})\lambda T$, with probability at least $1-\epsilon_2$, $\forall s \in [0,\ldots,S]$,

$$\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s) \geq (1-3\mathcal{F}_s)P^*_{s+1}(\mathbf{x}_{s+1}),$$

where $\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)$ is the objective value of P_{s+1} in (10) achieved by our solution $x_{i\eta}(\mathbf{p}_s)$, and $P^*_{s+1}(\mathbf{x}_{s+1})$ is the optimal objective value of P_{s+1} in (10) under optimal solution $\mathbf{x}_{s+1}$, given $\frac{c_k}{r^k_{max}} \geq \mathcal{B}$.

Proof: Please refer to Appendix D. □

Lemma 11: $E[P^*_s(\mathbf{x}_s)] \leq 2^s\epsilon_2 P^*_\Sigma, \forall s \in [0,\ldots,S]$, where $E(P^*_s(\mathbf{x}_s))$ is the expectation of the optimal objective value of P_s in (10) achieved by the optimal solution $\mathbf{x}_s$ over all possible realizations of the job arrival process, and P^*_Σ is the optimal objective value of (8).

Proof: Let $(\mathbf{x}_s, \mathbf{p}_s, \mathbf{u}_s)$ denote the optimal primal-dual solution to (10) and (11), and $(\mathbf{x}_\Sigma, \mathbf{p}_\Sigma, \mathbf{u}_\Sigma)$ denote the optimal primal-dual solution to (8) and (9). Comparing the two dual programs (11) and (9), we can observe that $(\mathbf{p}_\Sigma, \mathbf{u}_\Sigma)$ is a feasible solution to program D_s in (11) as any realization of job $i \in \mathcal{I}_s$ can be found in the distribution $\mathcal{D}$. Then the objective value of (11) with solution $(\mathbf{p}_\Sigma, \mathbf{u}_\Sigma)$, $D_s(\mathbf{p}_\Sigma, \mathbf{u}_\Sigma)$, is at least the optimal objective value $D^*_s(\mathbf{p}_s, \mathbf{u}_s)$. Furthermore, according to weak duality, $P^*_s(\mathbf{x}_s) \leq D^*_s(\mathbf{p}_s, \mathbf{u}_s) \leq D_s(\mathbf{p}_\Sigma, \mathbf{u}_\Sigma)$. Then we have

$$\begin{aligned}
&E[P^*_s(\mathbf{x}_s)] \leq E[D^*_s(\mathbf{p}_s, \mathbf{u}_s)] \leq E[D_s(\mathbf{p}_\Sigma, \mathbf{u}_\Sigma)]\\
&= E[\sum_{k\in[K]} (1-\mathcal{F}_s)2^s\epsilon_2 c_k p_{k,\Sigma} + \sum_{i\in\mathcal{I}_s} u_{i,\Sigma}]\\
&\leq E[2^s\epsilon_2 \sum_{k\in[K]} c_k p_{k,\Sigma} + \sum_{i\in\mathcal{I}_s} u_{i,\Sigma}]\\
&\leq 2^s\epsilon_2 \sum_{k\in[K]} c_k p_{k,\Sigma} + \sum_{j\in\mathcal{D}} I_s\rho_j u_{j,\Sigma}\\
&\leq 2^s\epsilon_2(\sum_{k\in[K]} c_k p_{k,\Sigma} + \sum_{j\in\mathcal{D}} \lambda T\rho_j u_{j,\Sigma})\\
&= 2^s\epsilon_2 D^*_\Sigma(\mathbf{p}_\Sigma, \mathbf{u}_\Sigma) = 2^s\epsilon_2 P^*_\Sigma.
\end{aligned}$$

□

Lemma 12: On the condition of $(1-\frac{\mathcal{F}_s}{2})\lambda T \leq I \leq (1+\frac{\mathcal{F}_s}{2})\lambda T$, $(1-\epsilon_2)P^*_\Sigma \leq E[P^*_{S+1}(\mathbf{x}_{S+1})] \leq (1+\mathcal{F}_s/2)P^*_\Sigma$ where $S+1 = \log_2(\frac{1}{\epsilon_2})$, $E(P^*_{S+1}(\mathbf{x}_{S+1}))$ is the expectation of the optimal objective value of P_{S+1} and P^*_Σ is the optimal objective value of (8).

Proof: We first prove that $E[P^*_{S+1}(\mathbf{x}_{S+1})] \leq (1+\mathcal{F}_s/2)P^*_\Sigma$. Similar to the proof in Lemma 11, we have

$$\begin{aligned}
&E[P^*_{S+1}(\mathbf{x}_{S+1})] \leq E[D^*_{S+1}(\mathbf{p}_{S+1},\mathbf{u}_{S+1})] \leq E[D_{S+1}(\mathbf{p}_\Sigma,\mathbf{u}_\Sigma)]\\
&\leq E[\sum_{k\in[K]} c_k p_{k,\Sigma} + \sum_{i\in[I]} u_{i,\Sigma}]\\
&\leq \sum_{k\in[K]} c_k p_{k,\Sigma} + \sum_{j\in\mathcal{D}}(1+\frac{\mathcal{F}_s}{2})\lambda T\rho_j u_{j,\Sigma}\\
&\leq (1+\frac{\mathcal{F}_s}{2})(\sum_{k\in[K]} c_k p_{k,\Sigma} + \sum_{j\in\mathcal{D}}\lambda T\rho_j u_{j,\Sigma})\\
&= (1+\frac{\mathcal{F}_s}{2})D^*_\Sigma(\mathbf{p}_\Sigma,\mathbf{u}_\Sigma) = (1+\frac{\mathcal{F}_s}{2})P^*_\Sigma.
\end{aligned}$$

Next, we show $(1-\epsilon_2)P^*_\Sigma \leq E[P^*_{S+1}(\mathbf{x}_{S+1})]$. When $S+1=\log_2(\frac{1}{\epsilon_2})$, constraints (10a) in program P_{S+1} becomes $\sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]}\frac{\omega^k_{i\eta}}{T}x_{i\eta} \leq (1-\epsilon_2)c_k$. Consider a new version of LP (8) by replacing constraints (8a) with $\sum_{j\in\mathcal{D}}\sum_{\eta\in[N_j]}\lambda T\rho_j\frac{\omega^k_{j\eta}}{T}x_{j\eta} \leq (1-\epsilon_2)c_k$, and denote this new program by $P_{\Sigma'}$. Let $P^*_{\Sigma'}$ be the optimal objective value of $P_{\Sigma'}$, and $\mathbf{x}_\Sigma$ be the optimal solution of (8). Then $(1-\epsilon_2)\mathbf{x}_\Sigma$ must be a feasible solution to $P_{\Sigma'}$, and the objective value under this solution is at most $P^*_{\Sigma'}$, *i.e.,* $P_{\Sigma'}(\mathbf{x}_\Sigma) = (1-\epsilon_2)P^*_\Sigma \leq P^*_{\Sigma'}$. In addition, compare P_{S+1} and $P_{\Sigma'}$, we found the expectation of optimal objective value of P_{S+1} is equal to $P^*_{\Sigma'}$. Therefore, $E[P^*_{S+1}(\mathbf{x}_{S+1})] = P^*_{\Sigma'} \geq (1-\epsilon_2)P^*_\Sigma$. □

Theorem 4: For any $0<\epsilon_2<1$, our online scheduling algorithm A_{online} is $(1-23\epsilon_2)$-competitive in expectation with *i.i.d.* job types and uniform job arrival time distribution, as compared to the expected optimal objective value of offline problem in (2), given $\lambda T \geq \frac{4}{(\epsilon_2)^3}$ and $\frac{c_k}{r^k_{max}} \geq \mathcal{B}$.

Proof: Combining Lemma 5, Lemma 7, and Lemma 10, we have with probability at least $(1-\epsilon_2)\times(1-\epsilon_2)\times(1-2\epsilon_2) \geq 1-4\epsilon_2$, events $(1-\frac{\mathcal{F}_s}{2})\lambda T \leq I \leq (1+\frac{\mathcal{F}_s}{2})\lambda T$, $\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s}E[\mathbf{Y}^k_i(t)] \leq 2^s\epsilon_2 c_k$, $\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s) \geq (1-3\mathcal{F}_s)P^*_{s+1}(\mathbf{x}_{s+1})$, happen simultaneously for all $k\in[K], t\in[T]$ and $s\in[0,\ldots,S]$. Let Ψ denote the event that three events happen simultaneously. Then we can have:

$$\begin{aligned}
&E[\sum_{s=0}^{S}\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)|\Psi]\\
&\geq E[\sum_{s}\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)|\Psi]\\
&\quad - E[\sum_{s}\sum_{i\in\mathcal{I}_{s}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)|\Psi]\\
&\geq \sum_s(1-3\mathcal{F}_s)E[P^*_{s+1}(\mathbf{x}_{s+1})|\Psi] - \sum_s E[P^*_s(\mathbf{x}_s)|\Psi] \qquad (17)
\end{aligned}$$

Combining Lemma 11 and Lemma 12, we have

$$\begin{aligned}
(17) &\geq (1-\epsilon_2)P^*_\Sigma - \frac{1}{Pr[\Psi]}\left(E[P^*_0(\mathbf{x}_0)] + \sum_s 3\mathcal{F}_s E[P^*_{s+1}(\mathbf{x}_{s+1})]\right)\\
&\geq (1-\epsilon_2)P^*_\Sigma - \frac{1}{1-4\epsilon_2}\left(\epsilon_2 + \sum_{s=0}^{S-1}3\epsilon\mathcal{F}_s 2_2^{s+1} + 3\mathcal{F}_S(1+\frac{\mathcal{F}_s}{2})\right)P^*_\Sigma\\
&\geq (1-\epsilon_2)P^*_\Sigma - \frac{1}{1-4\epsilon_2}(1+6\times 1.8+3\times\sqrt{2}\times(1+\frac{\sqrt{0.5}}{2})\epsilon_2 P^*_\Sigma\\
&\geq (1-\epsilon_2)P^*_\Sigma - \frac{1}{1-4\epsilon_2}18\epsilon_2 P^*_\Sigma.
\end{aligned}$$

The last two inequalities hold because $\sum_{s=0}^{S-1}\mathcal{F}_s 2^s\epsilon_2 \leq 1.8\epsilon_2, \mathcal{F}_s \leq \sqrt{\epsilon_2} \leq \sqrt{0.5}$.

$$\begin{aligned}
&E[\sum_{s=0}^{S}\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)|\Psi]\\
&\geq Pr[\Psi]\times E[\sum_{s=0}^{S}\sum_{i\in\mathcal{I}_{s+1}\setminus\mathcal{I}_s}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)]\\
&\geq (1-4\epsilon_2)\left((1-\epsilon_2)P^*_\Sigma - \frac{1}{1-4\epsilon_2}18\epsilon_2 P^*_\Sigma\right)\\
&\geq (1-23\epsilon_2)P^*_\Sigma \geq (1-23\epsilon_2)E[OPT].
\end{aligned}$$ □

D. Discussions

A_{online} can be generalized to handle general jobs with arbitrary dependence graph topology. Upon the arrival of the ith job, we first compute a topological ordering of its dependence graph. Such ordering ensures that if job i's subtask j must be executed before subtask k, j precedes k in the ordering. It can be accomplished in linear time, *e.g.,* by Kahn's algorithm or depth-first search [30]. We then re-index its subtasks according to the output ordering. The rest of the algorithm design is the same as the counterpart in Sec. V-B, and we omit the details. Because the expected offline optimization problem for general jobs can also be formulated to LP (8) and our online algorithm design is based on this LP, the online algorithm for general jobs can achieve the same performance as A_{online} does, with regard to optimality and feasibility. The theoretical analysis is similar to the counterpart in Sec. V-C, and is omitted here.

VI. Performance Evaluation

In this section, we evaluate our offline and online scheduling algorithms through trace-driven simulation studies. We further compare our scheduling algorithms with two related algorithms from recent literature [19] [37]. They study the similar cloud scheduling problem under simplified offline and online scenarios by assuming that each job contains only one subtask. We first introduce the simulation setup. We configure each job according to Google Cluster Date (version 1 [1]) which contains each job's information including number of subtasks, execution duration, and resource demands (CPU and RAM). We assume each subtask occupies $[1,12]$ slots, and each slot is 5 minutes. By default, the maximum number of subtasks (N) is 5, $\lambda = 0.5$ and $T = 500$. The total number of jobs I is decided according to a Poisson distribution with expectation of λT. The arrival time of each job is independently and uniformly chosen within $[1,T]$ to simulate a Poisson process. Each job's deadline is also generated uniformly at random between its arrival time and T. The value of each subtask (b_{in}) is computed as: its overall resource demand times unit prices randomly picked in the range $[1,50]$. The capacity of each type of resource is normalized to 1. The default value of $C = \min_{k\in[K]}\{\frac{c_k}{r^k_{max}}\}$ in our experiments is 1, which is much smaller than the value in our assumption. Although a lower bound of C is required for our theoretical analysis, it can be observed that even when the assumption is violated,

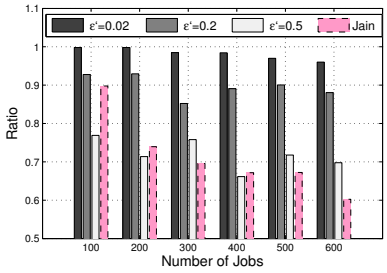


Fig. 2. Performance ratio of $A_{offline}$, and Jain *et al.*'s algorithm [19].

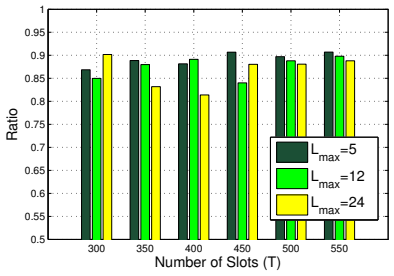


Fig. 3. Performance ratio of $A_{offline}$ with different T and L_{max}.

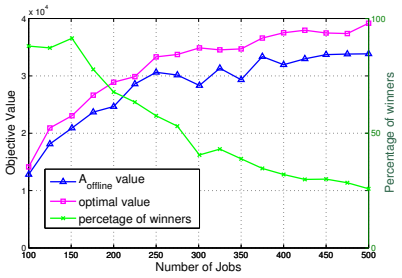


Fig. 4. $A_{offline}$: objective value and percentage of winners.

our offline and online algorithms can still achieve a close-to-optimal performance in practice.

A. Performance of $A_{offline}$

Performance Ratio. We first examine the performance of our offline algorithm, measured by the ratio of the average objective value of ILP (1) generated by $A_{offline}$ to the optimal objective value of ILP (1). The average objective value is obtained by running lines 2-6 in Algorithm 2 20 times. We also implement Jain *et al.*'s offline algorithm [19], which proposes a greedy strategy to select winners, for comparison with $A_{offline}$. Fig. 2 shows that the performance ratio of $A_{offline}$ decreases slightly when we increase the total number of jobs. In addition, the ratio is inversely related to the input parameter ϵ' to Algorithm 2, as confirmed by the analysis in Theorem 2. $A_{offline}$ achieves a close-to-optimal performance with a small ϵ' (0.02) and has a better performance than Jain *et al.*'s algorithm even when ϵ' is relatively large (0.2). We next fix ϵ' to 0.2 and the number of jobs to 300, and vary the number of slots and the maximum length of subtasks. Fig. 3 illustrates that both T and $L_{\max}$ have relatively small impact on the performance of $A_{offline}$. This is because our offline solution is derived from the fractional solution rather than the input of the problem.

Objective Value, Winner Satisfaction. and Time Complexity. Fig. 4 compares the objective value produced by $A_{offline}$ to the optimal value. Again, there is just a small gap between theses two values. The objective value grows with the increase of number of jobs because $A_{offline}$ can select more high-value jobs from a large set of jobs. The performance of $A_{offline}$ in terms of winner satisfaction, as measured by the percentage of winning jobs, is also demonstrated in Fig. 4. The percentage of winners drops when there is a large number of jobs. This is because the number of winners is relatively fixed and is limited by the resource capacity. Therefore, only a small percentage of jobs can be served from a large set of jobs. Next, we apply the `tic` and `toc` functions in MATLAB to measure the execution time of the main program without counting the initialization stage. We run 20 tests on a laptop computer (Intel Core i7-6700HQ/16GB RAM) and present the average result in Fig. 5. We can observe that the running time of $A_{offline}$ remains at a low level (< 20 seconds) even when we input a large number of jobs and a long time span. It increases linearly with the increases of jobs and slots, and runs faster than the theoretical result indicated in Lemma 2.

B. Performance of A_{online}

Performance Ratio. The expected offline objective value is estimated by exactly solving ILP (1) 20 times under different realization of the bid arrival process. The performance ratio of A_{online} is the ratio of the average objective value produced by A_{online} (over different realization of the bid arrival process) to the expected offline objective value. Fig. 6 shows that a better performance ratio comes with a smaller ϵ_2, while the arrival rate λ doesn't affect the ratio much. Comparing to the the performance ratio of $A_{offline}$ in Fig. 2, we observe that both A_{online} and $A_{offline}$ can achieve a close-to-optimal performance and our online algorithm performs slightly worse than our offline algorithm as it doesn't have access to future job information. In the following figures, we fix the value of ϵ_2 to 0.02 and examine the impact of other parameters. We vary the total number of slots, use the estimated λ as input to A_{online} and plot the performance ratio in Fig. 7. We observe that the ratio remains relatively steady with the growth of T. Over-estimation causes a worse performance than under-estimation, as compared to the real λ (labelled by 100%). This is because A_{online} rejects more jobs with an over-estimated λ. The good news is the ratio is still close to 0.9 even when we input an inaccurate λ.

We further compare our online algorithm with Zhou *et al.*'s online algorithm [37], which also conducts job admission based on the current resource prices. Their price is a function of U_k/L_k, where U_k (L_k) is the maximum (minimum) value per unit of type-k resource per unit of time. Fig. 8 and Fig. 9 show that A_{online} consistently outperforms Zhou *et al.*'s online algorithm over a wide range of U_k/L_k and number of slots (T). In Fig. 9, we set ϵ_2 to 0.2 and still observe the superiority of our online algorithm.

Objective Value and Winner Satisfaction. Next, we investigate the performance of A_{online}, in the aspects of achieved objective value and winner satisfaction. In Fig. 10, there is a upward trend in the objective value with the increment of the number of jobs. When ϵ_2 decreases, the solution output by A_{online} is closer to optimum, leading to a higher overall obtained value. Fig. 11 reflects that the percentage of winners also goes down with the increase of the number of jobs, similar to that of $A_{offline}$. Moreover, more jobs can be severed when the number of subtasks in each job raises since there is larger selection space for each job's execution.

Time Complexity and Feasibility. We test the running time of A_{online} under different input scales, and plot the result

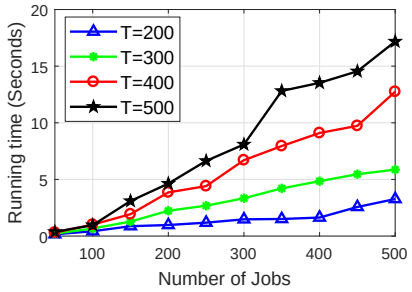


Fig. 5. Running time of $A_{offline}$ under different I and T.

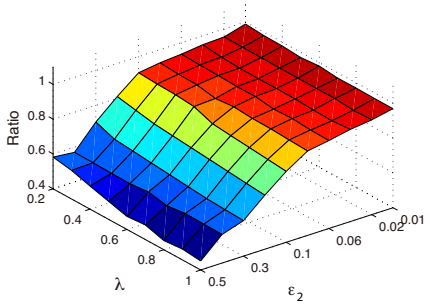


Fig. 6. Performance ratio of A_{online} under different λ and ϵ_2.

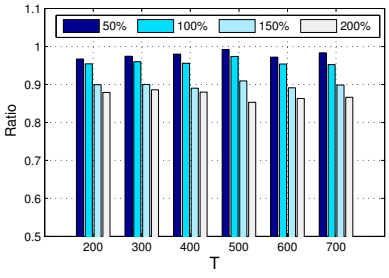


Fig. 7. Performance ratio of A_{online} with different estimations of λ under different T.

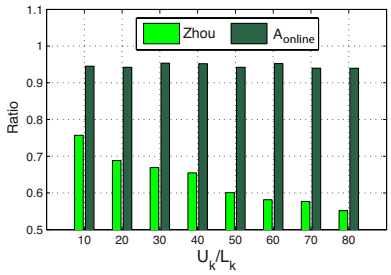


Fig. 8. Comparison between A_{online} and Zhou *et al.*'s online algorithm [37] under different U_k/L_k.

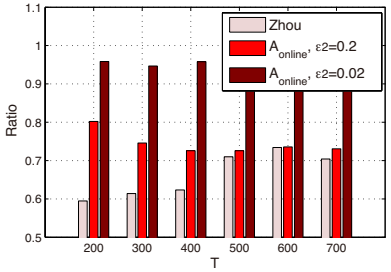


Fig. 9. Comparison between A_{online} and Zhou under different T.

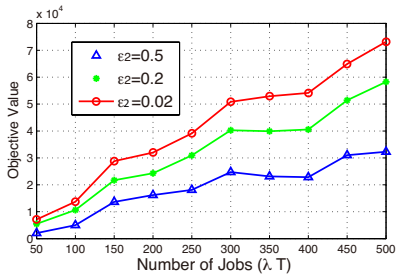


Fig. 10. Objective value achieved by A_{online}.

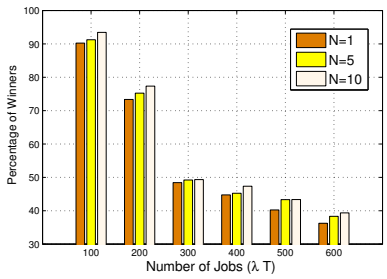


Fig. 11. Percentage of winners in A_{online}.

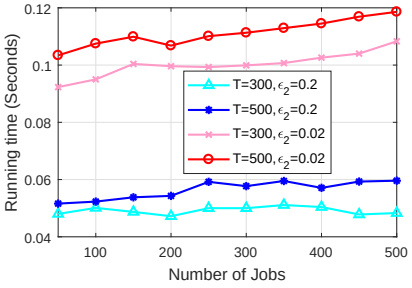


Fig. 12. Running time of A_{online} under different I, T, and ϵ_2.

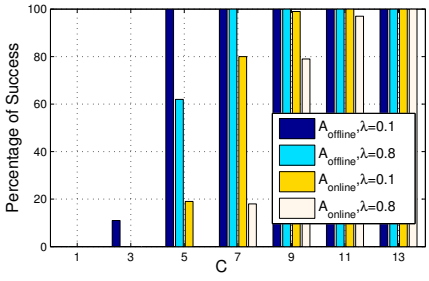


Fig. 13. Feasibility test for $A_{offline}$ and A_{online} when $T = 500$.

in Fig. 12. We can see that the worst case running time of A_{online} is shorter than 0.12 second, which is much smaller than that of $A_{offline}$. Moreover, its runtime slightly increases with the growths of the number of jobs and the number of time slots. The value of ϵ_2 determines its runtime. This is because ϵ_2 is used to compute the number of times to solve dual LP (11). Finally, we run feasibility test for $A_{offline}$ and A_{online}. In Theorem 2 and Lemma 7, we proved that with high probability, both $A_{offline}$ and A_{online} can produce feasible solutions. Therefore, we vary the value of C and the number of jobs (determined by λT). We run each algorithm 100 times, and count the number of successes, *i.e.,* the number of feasible solutions returned. As shown in Fig. 13, although we require C to be a large number in the theoretical proof, our simulation results show that both algorithms work well when $C > 10$, In addition, a larger number of jobs results in a lower success rate.

VII. Conclusion

We presented scheduling frameworks for cloud container services under both offline and online settings. Our problem model is expressive enough to accommodate complex cloud computing jobs. Our offline and online algorithms achieve computational and economical efficiencies. A natural direction for future research is to study the offline scheduling problem for general directed acyclic dependence graphs. It may also be promising to extend the application of our framework to other scheduling problems, *e.g.*, in 5G networks and smart grids.

References

[1] *Google Cluster Data, Version 1*. https://github.com/google/cluster-data/blob/master/TraceVersion1.md.
[2] S. Agrawal and N. R. Devanur. Fast algorithms for online stochastic convex programming. In *Proc. of ACM-SIAM SODA*, 2015.
[3] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014.
[4] Aliyun. *Container Service*. https://goo.gl/CnLfBQ.
[5] Amazon. *Amazon EC2 Container Service*. https://aws.amazon.com/ecs/.

[6] A. Archer, C. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Mathematics*, 1(2):129–150, 2004.

[7] Y. Azar, I. Kalp-Shaltiel, B. Lucier, I. Menache, J. S. Naor, and J. Yaniv. Truthful online scheduling with commitments. In *Proc. of ACM EC*, 2015.

[8] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang. On the competitiveness of on-line real-time task scheduling. In *Proc. of IEEE RTSS*, 1991.

[9] S. Brahma. *The Ellipsoid Algorithm for Linear Programming*. https://goo.gl/ge0p6u.

[10] E.-C. Chang and C. Yap. Competitive online scheduling with level of service. In *Proc. of International Computing and Combinatorics Conference*. Springer, 2001.

[11] F. Y. Chin and S. P. Fung. Improved competitive algorithms for online scheduling with partial job values. In *Proc. of International Computing and Combinatorics Conference*. Springer, 2003.

[12] F. Y. Chin and S. P. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

[13] M. Chrobak, L. Epstein, J. Noga, J. Sgall, R. van Stee, T. Tichỳ, and N. Vakhania. Preemptive scheduling in overloaded systems. In *International Colloquium on Automata, Languages, and Programming*, pages 800–811. Springer, 2002.

[14] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proc. of ACM-SIAM SODA*, 2006.

[15] Google. *Container Engine*. https://cloud.google.com/container-engine/.

[16] S. Gu, Z. Li, C. Wu, and C. Huang. An efficient auction mechanism for service chains in the nfv market. In *Proc. of IEEE INFOCOM*, 2016.

[17] A. Gupta and M. Molinaro. How the experts algorithm can help solve LPs online. *Mathematics of Operations Research*, 41(4):1404–1431, 2016.

[18] P. Jaillet and X. Lu. Near-optimal online algorithms for dynamic resource allocation problems. *arXiv:1208.2596*, 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):3, 2015.

[20] T. Kesselheim, A. Tönnis, K. Radke, and B. Vöcking. Primal beats dual on online packing LPs in the random-order model. In *Proc. of ACM STOC*, 2014.

[21] G. Koren and D. Shasha. Dˆover: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. *SIAM Journal on Computing*, 24(2):318–339, 1995.

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

[23] Microsoft. *Azure Container Service*. https://azure.microsoft.com/en-us/services/container-service/.

[24] Microsoft. *Batch feature overview for developers*. https://goo.gl/bQql24.

[25] P. Orlik and H. Terao. *Arrangements of hyperplanes*, volume 300. Springer Science & Business Media, 2013.

[26] P. Raghavan and R. Motwani. *Randomized Algorithms*. Cambridge Univ. Press, 1995.

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

[28] Wikipedia. *Karmarkar's algorithm*. https://en.wikipedia.org/wiki/Karmarkar's_algorithm.

[29] Wikipedia. *Poisson point process*. https://en.wikipedia.org/wiki/Poisson_point_process.

[30] Wikipedia. *Rendering Pipeline Overview*. https://www.opengl.org/wiki/Rendering_Pipeline_Overview.

[31] X. Xu, H. Yu, and X. Pei. A novel resource scheduling approach in container based clouds. In *Proc. of IEEE CSE*, 2014.

[32] ZDNet. *Containers: Fundamental to the cloud's evolution*. https://goo.gl/PPWmxe.

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

[34] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. Lau. Online auctions in IaaS clouds: welfare and profit maximization with server costs. In *Proc. of ACM SIGMETRICS*, 2015.

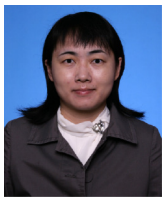
[35] Y. Zheng, B. Ji, N. Shroff, and P. Sinha. Forget the deadline: Scheduling interactive applications in data centers. In *Proc. of IEEE Cloud*, 2015.

[36] Z. Zheng and N. B. Shroff. Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud. In *Proc. of IEEE INFOCOM*, 2016.

[37] R. Zhou, Z. Li, C. Wu, and Z. Huang. An efficient cloud market mechanism for computing jobs with soft deadlines. *IEEE/ACM Transactions on Networking*, 25(2):793–805, 2017.

Ruiting Zhou received a B.E. degree in telecommunication engineering from Nanjing University of Post and Telecommunication, China, in 2007, a M.S. degree in telecommunications from Hong Kong University of Science and Technology, Hong Kong, in 2008 and a M.S. degree in computer science from University of Calgary, Canada, in 2012. Since March, 2016, she has been a PhD candidate at the Department of Computer Science, University of Calgary, Canada. Her research interests include smart grids, cloud computing and mobile network optimization.

Zongpeng Li received his B.E. degree in CS from Tsinghua University in 1999, his M.S. degree in CS from University of Toronto in 2001, and his Ph.D. degree in ECE from University of Toronto in 2005. Since then, Zongpeng has been a faculty member at the University of Calgary and Wuhan University. His research interests are in computer networks, network coding, cloud computing, and energy networks. Zongpeng was named an Edward S. Rogers Sr. Scholar in 2004, won the Alberta Ingenuity New Faculty Award in 2007, and was nominated for the Alfred P. Sloan Research Fellow in 2007. Zongpeng co-authored papers that received Best Paper Awards at the following conferences: PAM 2008, HotPOST 2012, and ACM e-Energy 2016. Zongpeng received the Department Excellence Award from the Department of Computer Science, University of Calgary, the "Outstanding Young Computer Science Researcher" Prize from the Canadian Association of Computer Science, and the Research Excellence Award (Early Career) from the Faculty of Science, University of Calgary.



Chuan Wu received her B.Engr. and M.Engr. degrees in 2000 and 2002 from the Department of Computer Science and Technology, Tsinghua University, China, and her Ph.D. degree in 2008 from the Department of Electrical and Computer Engineering, University of Toronto, Canada. Between 2002 and 2004, She worked in the Information Technology industry in Singapore. Since September 2008, Chuan Wu has been with the Department of Computer Science at the University of Hong Kong, where she is currently an Associate Professor. Her current research is in the areas of cloud computing, distributed machine learning/big data analytics systems, network function virtualization, and data center networking. She is a senior member of IEEE, a member of ACM, and served as the Chair of the Interest Group on Multimedia services and applications over Emerging Networks (MEN) of the IEEE Multimedia Communication Technical Committee (MMTC) from 2012 to 2014. She is an associate editor of IEEE Transactions on Multimedia and ACM Transactions on Modeling and Performance Evaluation of Computing Systems. She has also served as TPC members and reviewers for various international conferences and journals. She was the co-recipient of the best paper awards of HotPOST 2012 and ACM e-Energy 2016.

APPENDIX

A. Proof of Lemma 1

Line 1 takes $O(KN_i(d_i - a_i))$ steps to calculate the price in each time slot. The first `for` loop iterates N_i times and the second `for` loop iterates at most N_i times. Within the second `for` loop, lines 4-9 include two `for` loops to select the best schedule within a given time period and compute its price, which can be done in $O((d_i - a_i)^3 L_{in})$ steps as the execution time in line 6 is $O((d_i - a_i)L_{in})$. Lines 10-16 update the schedule and its price, taking $O((d_i - a_i)^3)$ steps. Therefore, the execution time for the second `for` loop (lines 3-17) is $O(N_i(d_i - a_i)^3 L_{\max})$ with $L_{\max} = \max_{i\in[I],n\in[N_i]}\{L_{in}\}$. Lines 18-19 takes $O((d_i - a_i)^2)$ steps to compute the best schedule. Hence, the running time from line 2 to 20 is $O(N_i^2(d_i - a_i)^3 L_{\max})$. The `if` statement in lines 21-25 returns the final output within $O(N_i)$ steps. In summary, the overall running time of Algorithm 1 is $O(KN_i^2(d_i - a_i)^3 L_{\max})$. □

B. Proof of Theorem 3

We first examine the running time of A_{core}. Lines 1-4 takes $O(N_i K)$ steps to compute u_i. The running time of the `if` statement in lines 5-15 is $O(\sum_{n\in[\eta_i]} L_{in})$. In summary, the running time of A_{core} is $O(N_i K + \sum_{n\in[\eta_i]} L_{in})$.

We next analyze the running time of A_{online}. Lines 1-2 define and initialize variables in $O(1)$ steps. There are I jobs, and the running time to handle each job (lines 4-13) is dominated by the running time of A_{core}. The body of the `if` statement in lines 15-16 is executed $\lfloor \log_2(\frac{1}{\epsilon_2}) \rfloor$ times, each iteration solves the dual problem in (11) in $O(I^{3.5})\mathcal{L}$ steps using Karmarkar's algorithm [28], where each job is encoded in $\mathcal{L}$ bits. Recall that $N = \max_{i\in[I]}\{N_i\}$ and $L_{max} = \max_{i\in[I],n\in[N_i]} L_{in}$. Give the above, the time complexity of our online algorithm A_{online} is $O(\lfloor \log_2(\frac{1}{\epsilon_2}) \rfloor I^{3.5}\mathcal{L} + IN(K + L))$. □

C. Proof of Lemma 4

If job i is rejected by the cloud provider, *i.e.,* $x_{i\eta}(\mathbf{p}_s) = 0, \forall \eta \in [N_i]$, then $\mathbf{Y}_i^k(t) = \mathbf{X}_i^k$ and the lemma follows. Otherwise, let $\eta_i = \arg\max_{\eta\in[N_i]} x_{i\eta}(\mathbf{p}_s)$. According to A_{online}, job i is scheduled within $[a_i, a_i + \sum_{n=1}^{\eta_i} L_{in})$, and let l_i be the corresponding schedule. For a fixed $t \in [T]$, we have

$$E(\mathbf{Y}_i^k(t)) = Pr[a_i \le t < a_i + \sum_{n=1}^{\eta_i} L_{in})] f_{il_i}^k(t)$$
$$= Pr[t - \sum_{n=1}^{\eta_i} L_{in} < a_i \le t] f_{il_i}^k(t).$$

Because a_i is is uniformly disturbed in $[T]$, $Pr[a_i = t] = \frac{1}{T}$. Moreover, since $1 \le t \le T$, $Pr[t - \sum_{n=1}^{\eta_i} L_{in} < a_i \le t]$ has two different values when $1 \le t < \sum_{n=1}^{\eta_i} L_{in}$ and $\sum_{n=1}^{\eta_i} L_{in} \le t \le T$. For both cases, we have

$$E(\mathbf{Y}_i^k(t)) \le \frac{1}{T}\sum_{n=1}^{\eta_i} r_{in}^k L_{in} = \frac{\omega_{i\eta_i}^k}{T} = \mathbf{X}_i^k. \qquad \square$$

D. Proof of Lemma 10

We first define an auxiliary primal problem as follows:

$$P_A: \quad \text{maximize} \quad \sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta} \tag{18}$$

subject to:

$$\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T} x_{i\eta} \le A_k, \ \forall k \in [K], \tag{18a}$$

$$\sum_{\eta\in[N_i]} x_{i\eta} \le 1, \ \forall i \in \mathcal{I}_{s+1}, \tag{18b}$$

$$x_{i\eta} \ge 0, \forall i \in \mathcal{I}_{s+1}, \ \forall \eta \in [N_i]. \tag{18c}$$

where $A_k = \sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T} x_{i\eta}(\mathbf{p}_s)$ if $p_{k,s} > 0$ and $A_k = \max\{\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T} x_{i\eta}(\mathbf{p}_s), 2^{s+1}\epsilon_2 c_k\}$ if $p_{k,s} = 0$. Its dual problem is:

$$D_A: \quad \text{minimize} \quad \sum_{k\in[K]} A_k c_k p_k + \sum_{i\in\mathcal{I}_{s+1}} u_i \tag{19}$$

subject to:

$$u_i \ge b_{i\eta} - \sum_{k\in[K]} \frac{\omega_{i\eta}^k}{T} p_k, \ \forall i \in \mathcal{I}_{s+1}, \forall \eta \in [N_i], \tag{19a}$$

$$p_k, u_i \ge 0, \ \forall k \in [K], \forall i \in \mathcal{I}_{s+1}. \tag{19b}$$

Note that $\{x_{i\eta}(\mathbf{p}_s)\}_{i\in[i],\eta\in[N_i]}$ and $\mathbf{p}_s$ satisfy all complementarity conditions, and therefore they are the optimal primal and dual solutions to LP (18) and LP (19). The optimal objective value of (18) is $\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} b_{i\eta}x_{i\eta}(\mathbf{p}_s)$. In order to prove the lemma, we need show that with probability at least $1-\epsilon_2$, $(1-3\mathcal{F}_s)\mathbf{x}_{s+1}$ is a feasible solution to auxiliary program (18).

First, we show that with probability at least $1 - \epsilon_2$,

$$A_k \ge (1 - 3\mathcal{F}_s)2^{s+1}\epsilon_2 c_k, \forall k \in [K], s \in [0, \dots, S]. \tag{20}$$

If $p_{k,s} = 0$, then by definition we have $A_k \ge 2^{s+1}\epsilon_2 c_k$. It remains to prove the case where $p_{k,s} > 0$ that, with probability at least $1 - \epsilon_2$, $A_k \ge (1 - 3\mathcal{F}_s)2^{s+1}\epsilon_2 c_k, \forall k \in [K], s \in [0, \dots, S]$. This is proven by showing that with probability at most ϵ_2, $A_k = \sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T} x_{i\eta}(\mathbf{p}_s) \le (1 - 3\mathcal{F}_s)2^{s+1}\epsilon_2 c_k, \forall k \in [K], s \in [0, \dots, S]$. The detailed proof is as follows: Recall that $\{x_{i\eta,s}\}_{i\in[I],\eta\in[N_i]}$ and $\{p_{k,s}\}_{k\in[K]}$ are the optimal solutions to programs (10) and (11). Then, by complementary slackness, if $p_{k,s} > 0$, we have $\sum_{i\in\mathcal{I}_{s+1}} \sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^{k,s}}{T} x_{i\eta,s} = (1 - \mathcal{F}_s)2^s\epsilon_2 c_k$. We normalize r_{max}^k such that $r_{max}^k = 1$. Given $c_k/r_{max}^k \ge \frac{4\lambda T}{\epsilon_2^2} \ge \frac{\lambda T}{2^s\epsilon_2^2}$, and the observation in Lemma 9, we have for any k and s,

$$\sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T} x_{i\eta}(\mathbf{p}_s) \ge \sum_{i\in\mathcal{I}_s}\sum_{\eta\in[N_i]} \frac{\omega_{i\eta}^k}{T} x_{i\eta,s} - \lambda T$$
$$\ge (1 - \mathcal{F}_s - \epsilon_2)2^s\epsilon_2 c_k \ge (1 - 2\mathcal{F}_s)2^s\epsilon_2 c_k.$$

For a fixed k, s and a distinct price vector $\mathbf{p}$, when $\mathbf{p} = \mathbf{p}_s$, we define events $G_1 = \{\sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}_i^k \le (1 - 3\mathcal{F}_s)2^{s+1}\epsilon_2 c_k\}$ and $G_2 = \{\sum_{i\in\mathcal{I}_s} \mathbf{X}_i^k \ge (1 - 2\mathcal{F}_s)2^s\epsilon_2 c_k\}$.

$$Pr[G_1] = Pr[G_1|G_2] \le Pr[|\sum_{i\in\mathcal{I}_s} \mathbf{X}_i^k - \frac{I_s}{I_{s+1}} \sum_{i\in\mathcal{I}_{s+1}} \mathbf{X}_i^k| \ge \beta']. \tag{21}$$

Because $\frac{I_s}{I_{s+1}} = \frac{1}{2} \le \frac{1}{2(1-\mathcal{F}_s/2)}$ or $\frac{I_s}{I_{s+1}} = \frac{\lambda T/2}{I} \le \frac{1}{2(1-\mathcal{F}_s/2)}$

as $I \geq (1-\frac{\mathcal{F}_s}{2})\lambda T$, thus,

$$\sum_{i\in\mathcal{I}_s}\mathbf{X}_i^k - \frac{I_s}{I_{s+1}}\sum_{i\in\mathcal{I}_{s+1}}\mathbf{X}_i^k \geq$$
$$(1-2\mathcal{F}_s - \frac{1}{1-\mathcal{F}_s/2}(1-3\mathcal{F}_s))2^s\epsilon_2 c_k$$
$$= \frac{\mathcal{F}_s+2\mathcal{F}_s^2}{2-\mathcal{F}_s}2^s\epsilon_2 c_k \geq \frac{\mathcal{F}_s}{2}2^s\epsilon_2 c_k.$$

Then $\beta' = \frac{\mathcal{F}_s}{2}2^s\epsilon_2 c_k$. Note that $\mathbf{X}_i^k \in [0,1]$ as $r_{max}^k = 1$. Next, similar to the proof of Lemma 6, we define two random variables:

$$\sigma^2(\mathbf{X}) = \frac{1}{I_{s+1}}\sum_{i\in\mathcal{I}_{s+1}}(\mathbf{X}_i^k - \frac{1}{I_{s+1}}\sum_{i\in\mathcal{I}_{s+1}}\mathbf{X}_i^k)^2 \leq 1.$$

$$\Delta(\mathbf{X}) = \max_{i\in\mathcal{I}_{s+1}}\mathbf{X}_i^k - \min_{i\in\mathcal{I}_{s+1}}\mathbf{X}_i^k \leq 1.$$

According to Hoeffding-Berstein Inequality [3], we have

$$(21) \leq 2\exp\left(-\frac{\beta'^2}{2I_s\sigma^2(\mathbf{X})+\beta\Delta(\mathbf{X})}\right)$$
$$\leq 2\exp\left(-\frac{\frac{\mathcal{F}_s^2}{4}2^{2s}\epsilon_2^2c_k^2}{2I_s+\frac{\mathcal{F}_s}{2}2^s\epsilon_2 c_k}\right). \quad (22)$$

Because $c_k/r_{max}^k = c_k \geq \lambda T$, we have $2I_s \leq 2\cdot 2^s\epsilon_2\lambda T \leq 2\cdot 2^s\epsilon_2 c_k$. Hence,

$$(22) \leq 2\exp\left(-\frac{\frac{\mathcal{F}_s^2}{4}2^s\epsilon_2 c_k}{2+\frac{\mathcal{F}_s}{2}}\right) \leq 2\exp\left(-\frac{\epsilon_2^2 c_k}{12}\right)$$
$$\leq \frac{\epsilon_2}{K(IN)^K\log_2(\frac{1}{\epsilon_2})}.$$

The last inequality holds because $c_k/r_{max}^k = c_k \geq \mathcal{B}$. Taking union bound over $(IN)^K$ distinct prices, K types of resources and $\log_2(\frac{1}{\epsilon_2})$ stages, we prove that with probability at least $1-\epsilon_2$, $A_k \geq (1-3\mathcal{F}_s)2^{s+1}\epsilon_2 c_k, \forall k \in [K], s \in [0,\ldots,S]$.

We observe that i) constraints (10b) and (10c) are the same as (18b) and (18c); ii) constraints (10a) and (18a) only differ in the RHS. Following the result of (20), we have with probability at least $1-\epsilon_2$, $(1-3\mathcal{F}_s)\mathbf{x}_{s+1}$ is a feasible solution to LP (18). Therefore, with probability at least $1-\epsilon_2$, the optimal objective value of (18), *i.e.*, $\sum_{i\in\mathcal{I}_{s+1}}\sum_{\eta\in[N_i]}b_{i\eta}x_{i\eta}(\mathbf{p}_s)$, is at least the objective value of (18) under the solution $(1-3\mathcal{F}_s)\mathbf{x}_{s+1}$, *i.e.*, $(1-3\mathcal{F}_s)P_{s+1}^*(\mathbf{x}_{s+1})$. □