Online Workload Scheduling for Social Welfare Maximization in the Computing Continuum

Hailiang Zhao, Ziqi Wang, Guanjie Cheng, Wenzhuo Qian, Peng Chen, Jianwei Yin, Schahram Dustdar, *Fellow, IEEE*, and Shuiguang Deng, *Senior Member, IEEE*

***Abstract*—Computing ecosystems are shifting toward a computing continuum paradigm designed to handle the diverse and dynamic nature of computing resources spread across various locations. It demonstrates significant potential in providing high-bandwidth and low-latency services for users. However, as a large number of users request services from distributed computing continuum systems, it is critical to schedule numerous delay-sensitive, fractional workloads and maximum parallelism-bound jobs to appropriate backend resources, *e.g.*, cloud container instances. In addition, the scheduling strategy also needs to maximize the social welfare that incorporates the utilities of jobs and the revenue of service providers. However, current workload scheduling algorithms are based on simple heuristics and lack performance guarantees. Due to the unpredictability of online requests, the distribution of requests should not be assumed. Therefore, designing an online workload scheduling strategy without assumptions on request distributions is essential for balancing the online workload. This work first establishes a spatiotemporal integrated resource pool to reflect the computational resources provided by distributed computing continuum systems. Then, several pseudo-social welfare functions and marginal cost functions are constructed, where the latter is used to estimate the marginal cost of provisioning services to each newly arrived job based on the current resource surplus. We propose an online workload scheduling strategy named `OnSocMax` to solve the above problems. It operates by following the solutions to several convex pseudo-social welfare maximization problems and is proven to be α-competitive for some α with a value of at least 2. The evaluation results demonstrate that `OnSocMax` outperforms several benchmark strategies in maximizing social welfare.**

***Index Terms*—Computing continuum, mobile edge computing, load balancing, social welfare maximization, online workload scheduling.**

I. Introduction

Nowadays, the computing continuum [1] provides a new paradigm for users with varying computing demands, *e.g.*, smart mobile devices (SMDs), intelligent campuses [2], and industrial internet of things [3]. The computing continuum integrates decentralized physical and virtual computing resources into a holistic environment. For instance, cloud-assisted mobile edge computing (CMEC) systems integrate cloud and edge resources and are designed to present the user as a single pool of resources. Thus, it can provide high bandwidth and low latency services for users [4]. In that case, CMEC supports applications such as digital twins, virtual reality, and cloud games, significantly enhancing factory intelligence and enriching people's daily lives. In the CMEC system, some users submit computational tasks to the system irregularly, *e.g.*, a factory requests services from the system to analyze the defective percentage based on the production line camera data. To reduce the tedious process of configuring the VM-based machines while supporting user business logic design, serverless is introduced by utilizing the resources in the CMEC system to handle the system administration operations virtually [5], including installing operating systems, libraries, and runtime dependencies. In addition, the *on-demand resource provisioning* paradigm enables the CMEC system to allocate and provide resources flexibly based on actual demand, avoiding users' paying for idle resources and improving resource utilization of the system.

However, when many online requests for system services, a fundamental challenge is to balance the system's workload [6]. The load balancer is a component that exposes the APIs to the workloads by mapping the requests to carefully selected backend resources. Theoretically, it decides how the workloads are dispatched to distributed computing instances and how to allocate restricted resources to them, aiming to optimize end users' service quality. Due to the characteristics of online optimization, workload dispatching should be decided without knowledge of job arrivals [7]. However, load balancing strategies such as *RoundRobin* and *SessionAffinity* are simple to operate but offer no performance guarantee. Meanwhile, academic studies on online load balancing promise long-term performance guarantees. They assume that jobs arrive according to the Poisson process and that service rates of computing instances are exponentially distributed [8]. Under stochastic ordering assumption, policies including Join-the-Shortest-Queue (JSQ), Join-the-Idle-Queue (JIQ), Power-of-d-Choices (Pod), and Join-the-Fastest-of-the-Shortest-Queues (JFIQ) are proposed based on Continuous-Time Markov Chains (CTMC) and Lyapunov Stability theories. However, their performance guarantees (mostly on mean response time) are established on sufficient assumptions, which are tough to satisfy in real-world systems. Further, if we consider realistic constraints, including heterogeneous service rates, service locality[1], and strict deadlines, the performance guarantees are even harder to achieve.

Hailiang Zhao, Ziqi Wang, and Guanjie Cheng are with the School of Software Technology, Zhejiang University. Emails: {hliangzhao, ziqiwang}@zju.edu.cn, guanjiech@126.com.

Wenzhuo Qian, Peng Chen, Jianwei Yin, and Shuiguang Deng are with the College of Computer Science and Technology, Zhejiang University. Emails: {qwz, pgchen, zjuyjw, dengsg}@zju.edu.cn.

Schahram Dustdar is with the Distributed Systems Group at the TU Wien and with ICREA at the UPF, Barcelona. Email: dustdar@dsg.tuwien.ac.at.

[1]Service locality means the required functions cannot be executed on the chosen resource unit because the runtime or dependencies are not satisfied.

Online workload scheduling for delay-sensitive jobs has been considered mainly from the perspective of jobs' utility maximization. Due to the *on-demand resource provisioning* paradigm [9], this work also considers the revenue of CMEC service providers. We focus on social welfare maximization, where the utilities of jobs and the revenue of the CMEC service providers are maximized simultaneously. Thus, this work studies the online workload scheduling problem for delay-sensitive, fractional workloads and maximum parallelism-bound jobs by maximizing the sum of utilities of both jobs and the service provider. Our model is based on the designed resources pool considering the spatiotemporal relationship of multiple coupled computational units in the computing continuum. Each arrived job with fractional workloads can only be dispatched to its available resource units decided by jobs' parallelism-bound and deadline constraints. An online workload scheduling algorithm named `OnSocMax` is proposed to decide how the input workloads of arrived jobs are partitioned and scheduled under realistic constraints, such as anti-affinity, service locality, and unpredictable system failures of computing instances, *etc*. In addition, we construct marginal cost functions to estimate the marginal cost of provisioning services to each newly arrived job based on the current resource surplus. `OnSocMax` works by solving several well-designed pseudo-social welfare maximization problems online and has no assumptions on the arrival pattern and service rates. Finally, this work provides rigorous analysis to show that `OnSocMax` is α-competitive for some $\alpha \geq 2$. The main contributions of this work are summarized as follows.

1) A spatiotemporal integrated resource pool and resource units are designed for the computing continuum. Based on this design, this work studies the online social welfare maximization problem for delay-sensitive, fractional workloads and maximum parallelism-bound jobs. The proposed model simulates real-world online workload scheduling in CMEC systems and only assumes the utility functions.
2) An online workload scheduling algorithm named `OnSocMax` is proposed to yield a competitive ratio of at least 2 for general utility settings. Notably, it has a linear complexity when the utility of jobs is linear and shares the same coefficient.
3) This work formulates the detailed expression for the marginal cost of each resource unit. It estimates the cost for provisioning services to each newly arrived job as a function of resource surplus. In addition, the requirements of the marginal cost functions are given to maintain the α-competitive for some underlying α of the algorithm.

The remainder of this work is structured as follows. Section II discusses the related work. Section III introduces the spatiotemporal integrated resource pool and formulates the online social welfare maximization problem. Section IV presents design details of the online workload scheduling algorithm `OnSocMax` with sufficient theoretical analysis. The numerical results of the experiments are shown in Section V. Finally, Section VI concludes this work.

II. Related Works

A. Resource and Job Scheduling in Computing Continuum

The computing continuum represents a dynamic ecosystem that spans from edge devices to cloud infrastructure, where the status of endpoint devices and edge nodes are dynamically changed [10]. To ensure optimal performance and reliability, the system must efficiently handle various services' scheduling, deployment, and management to optimize quality-of-service (QoS) requirements. CMEC is a crucial scenario in the computing continuum to enhance service response time and efficiency. In this system, efficient task scheduling and reasonable resource allocation can improve its QoS. Thus, it is aimed to develop intelligent resource allocation and task scheduling strategies based on different realistic factors, including computing power, resource surplus, and network conditions. Due to the limited computing resources of small base stations (SBSs) in the edge, service requests generally wait in a queue for execution. These service requests vary in terms of required resources and execution time. Therefore, proper job scheduling in the CMEC system is necessary to fully utilize limited resources.

In recent years, extensive research has focused on efficient scheduling algorithms for different objectives in response to complex online service requests. Bi *et al.* [11] focus on task scheduling and resource allocation in the CMEC system to minimize total system cost while meeting SMD's latency requirements. A two-stage metaheuristic approach named LSAG is proposed to solve this problem. Specifically, the first stage determines the optimal edge selection strategy for handling cases involving multiple available SBSs. The second stage jointly optimizes task scheduling and resource allocation in the system. Liao *et al.* [12] propose an electric vehicle-assisted edge computing architecture that leverages the idle computing resources of electric vehicles (EVS). The goal is to reduce the system's energy consumption by selecting appropriate EVs to handle service requests. The authors construct an energy-aware workload offloading model and discretize the original model into multiple solvable problems with small scales.

Liu *et al.* [13] and Chen *et al.* [14] focus on optimizing latency in computing continuum systems. The former proposes a deep learning-assisted online algorithm to optimize latency in heterogeneous MEC systems. A deep neural network is adopted to emulate an offline solver of this problem and produce near-optimal solutions. The latter minimizes the average latency of SMDs by jointly optimizing communication, computation, and caching configurations in the system. Chen *et al.* [15] note that making offloading and resource allocation decisions in mobile environments is challenging due to the dynamic nature of wireless channel conditions and user locations. To address these issues, the authors investigate service collaboration within SMD service chains, focusing on a master-slave dependency model. Their goal is to minimize the system's latency and energy consumption. Mohajer *et al.* [16] propose a dynamic optimization model to maximize energy efficiency and system throughput in a computing continuum system. The optimization problem is decomposed into computational carrier scheduling and resource allocation. The authors employ a subgradient method for computational resource allocation and successive convex approximation with dual decomposition methods to solve services' fairness problems. Du *et al.* [17] aims to maximize the social welfare of the CMEC system by optimizing the allocation of computational resources and pricing of computing services for each SBS. The authors propose a two-tier algorithm to solve this problem. Specifically, the first tier addresses the association between users and SBSs, while the second tier focuses on resource allocation among SBSs to effectively utilize limited computational resources.

B. Online Workload Scheduling with Load Balance

Optimizing resource allocation and achieving load balancing are essential in online workload scheduling. It refers to the dispatch of numerous computational tasks to appropriate backend resources of edge or cloud while ensuring the load balance of the system. Load balance needs to consider a combination of performance goals, capabilities, and constraints.

Online load balancing policies are investigated under classic settings, where multiple identical servers with exponentially distributed service rates process continuous arrived jobs. Based on continuous-time Markov chain and Lyapunov stability theories, load-balancing policies such as JSQ [18] and JIQ [19] are proposed and analyzed on the mean response time and cross-server communication overhead. However, JSQ lacks a server-side perspective, and when the same service instance is invoked by multiple clients, the results produced by the JSQ algorithm are not optimal. In addition, due to its centralized algorithmic design, it incurs high communication overhead during distributed scheduling. JIQ, as a distributed load-balancing algorithm, reduces communication between the scheduler and the processor by letting newly arriving tasks join the idle queue instead of choosing the shortest queue. Based on the above techniques, Weng *et al.* [20] propose join-the-fastest-of-the-shortest-queue (JFSQ) and JFIQ policies under heterogeneous service rates and service locality constraints. They show that, under a well-connected bipartite graph condition, these policies can achieve the minimum mean response time in both the many-server and the sub-Halfin-Whitt regimes. Wu *et al.* [21] develop a computational offloading model in an MEC environment based on mean-field game theory and introduce a mean-field game-based load-balancing algorithm. This approach aims to reduce processing latency and streamline task scheduling using multi-agent deep reinforcement learning techniques. Each SMD within the MEC system is envisioned as an active participant in the mean-field game, transforming the intricate stochastic game into a more tractable dual-agent game for effective workload balancing.

Another line of work studies the cost-efficient and multi-resource sharing load balancing from a different theoretical basis. Generally, the objective is to improve energy efficiency with on-demand resource allocation. Thereinto, online load balancing of delay-sensitive jobs is studied in [22]–[24]. Zheng *et al.* [25] design online algorithms for fractional and non-fractional workload models under concave utility settings. The optimality of designed algorithms holds when all the jobs have the same deadline and share a single resource type. Duan *et al.* [26] investigate the load balance under user mobilities features in MEC environments. The objective is to solve the mobility-conscious online task offloading problem that incorporates adaptive load balancing, aiming to minimize overall computational costs. However, the complexity of this problem is heightened by the unpredictability of future user mobility patterns and the spatiotemporal variability of edge server computation loads. Thus, the primary task offloading optimization problem is strategically divided into two subsidiary problems: task offloading control and server grouping. Then, a long short-term memory-based algorithm and a dueling double deep Q-network-based algorithm are designed to address these subsidiary problems, thereby enhancing load balance within the MEC system. Online load balancing to minimize the Nash Social Welfare is investigated in [27]. The authors highlight two essential challenges in load balancing. The first is selfish load balancing, characterized by clients acting as non-cooperative, self-interested entities focused solely on minimizing their costs. The second variant is online load balancing, which involves clients arriving online and necessitating irrevocable assignment to a resource without foresight into future service requests. In addition, this work provides tight bounds on the price of anarchy of pure Nash equilibria and the competitive ratio of general greedy algorithms under different latency functions.

Unlike the above works, this work is more general regarding the concept of the resource pool and the technique of marginal cost estimation, making it more suitable for computing continuum systems. We take a CMEC system as a typical example and aim to maximize the social welfare of the system while balancing its workload. Social welfare incorporates the utility of both users and service providers. The proposed online workload scheduling algorithm `OnSocMax` works by solving several well-designed pseudo-social welfare maximization problems online and has no assumptions on the arrival pattern and service rates. It considers allocating resources over longer time horizons and aims to maximize the cumulative social welfare over the entire time slot. For performance guarantees, `OnSocMax` is proved to be α-competitive for some $\alpha \geq 2$.

III. System Model and Problem Formulation

In the computing continuum paradigm, we consider a general CMEC environment composed of multiple computing instances distributed geographically across different SBSs and cloud data center (CDC) sites [28]. We build our model on serverless because it removes the need to explicitly provision and manage computing instances. It is shown in Fig. 1 that in a typical example of the computing continuum, CMEC environment, each end-user establishes communications with one or more SBSs by wired or wireless transmission. Users can request SBS services online if they are in its coverage area. Due to the limited computing capability of SBSs, CMEC leverages the abundant computing resources in the cloud. Thus, Some SBSs connect to the CDC via low-latency fiber links [29]. In this case, each user is served by an invisible resource pool from geographically diverse cloud container instances from SBSs or CDC. This section first constructs the spatiotemporal resource pool for users and then formulates the social welfare maximization problem.

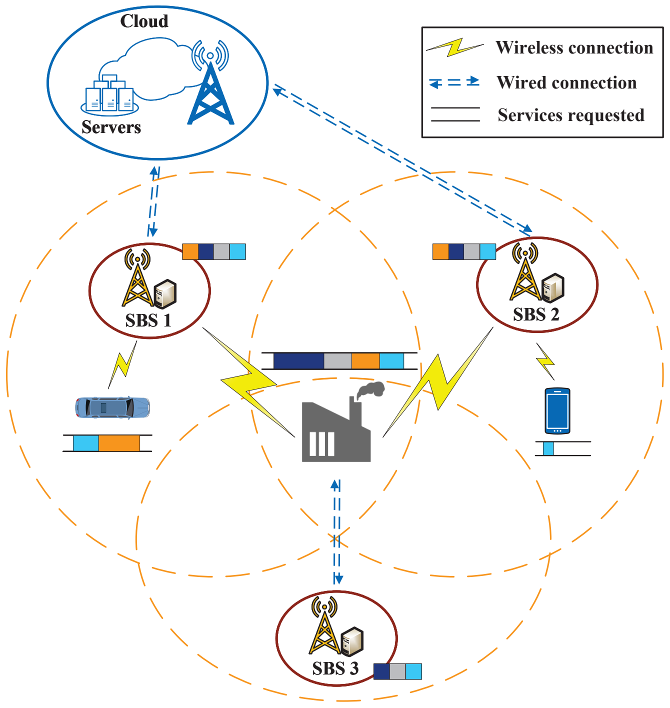


Fig. 1. Intensive online service requests in the CMEC system.

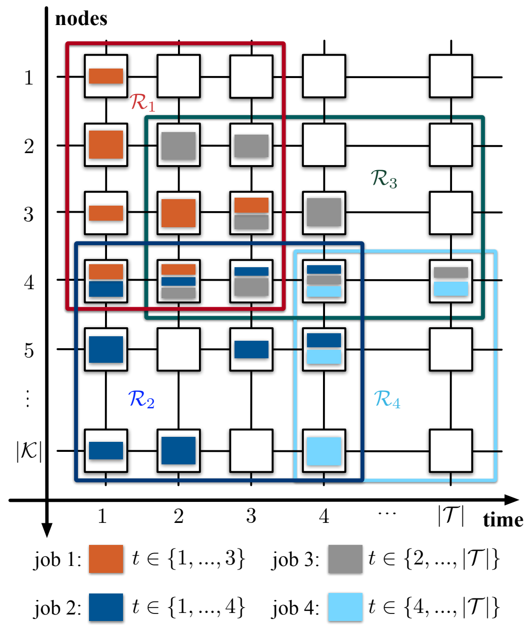

nodes
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_3$
$\mathcal{R}_4$
1
2
3
4
5
⋮
$|\mathcal{K}|$
1
2
3
4
⋯
$|\mathcal{T}|$
time
job 1: $t \in \{1, ..., 3\}$
job 2: $t \in \{1, ..., 4\}$
job 3: $t \in \{2, ..., |\mathcal{T}|\}$
job 4: $t \in \{4, ..., |\mathcal{T}|\}$

Fig. 2. Available resource units for four jobs in global resource mesh. Whether a resource unit r is available or not to job n is decided by both $\mathcal{R}_n$ and $\overline{\chi}_{nr}$.

A. Spatio-Temporal Resource Pool

Fig. 2 shows an example resource pool for an end user in the computing continuum paradigm. Let $\mathcal{K}$ denote the set of computing instances and index each of them by k. The CMEC system can process heterogeneous jobs arriving in sequence with different service rates. The set of jobs is denoted as $\mathcal{N}$ and indexes each job by n. Each job n has the input workloads of size ϱ_n. Jobs are processed by invoking cloud functions across different resource units, *e.g.* functions 1, 2, 3, and 4 in Fig. 2.

Take the video transcoding job [30] as an example. The inputs are raw video frames. It first partitions the input into frame pieces at a negligible cost. Then, it parallelizes the "slow" fragments of the encoding and performs the "fast" pieces serially. For discrete jobs unsuitable for parallelism, our model still applies with appropriate rounding policies, *e.g.*, the Fenchel duality [31] taken in [32]. $\forall n \in \mathcal{N}$, we use a_n and d_n to represent its arrival time and the deadline to be finished. To maximize the utilities of users and the revenue of the CMEC provider from a long-term vision, the time horizon is considered from $\min_{n\in\mathcal{N}} a_n$ to $\max_{n\in\mathcal{N}} d_n$ and evenly divide the horizon into slots of length τ. Let $\mathcal{T}$ denote the set of time slots and index each of them with t. The time slot length τ can be set as one-fourth of the minimum instance reserved time, for example, 15 minutes for AWS spot instance[2].

[2]https://aws.amazon.com/ec2/spot/pricing/

To manipulate the computing resources in $\mathcal{K}$ from both dimensions of time and space, we introduce a spatio-temporal resource division model named *resource pool*. $\mathcal{R} := \mathcal{K} \times \mathcal{T}$ denotes the set of resource units and indexes each of them by r. Each resource unit r can process at most $\mathcal{C}_r$ workloads of all jobs. This value could be obtained through various approaches, from static code analysis to profiling previous runs based on hardware heterogeneity [33]. For each job $n \in \mathcal{N}$, its available resource units must be available during its arrival time and deadline, *i.e.*,

$$\mathcal{R}_n := \left\{ r_{kt} \in \mathcal{R} \mid \left\lceil \frac{a_n}{\tau} \right\rceil \le t \le \left\lfloor \frac{d_n}{\tau} \right\rfloor, k \in \mathcal{K} \right\}. \tag{1}$$

B. Utility and Revenue Functions

This section introduces the utility of users and the revenue function of CMEC service providers. For each job $n \in \mathcal{N}$, we need to decide how to dispatch the workloads to its available resource units under parallelism limit and deadline constraint for maximizing the social welfare, which is the sum of all jobs' utilities and the revenue of the CMEC service providers. Formally, x_{nr} is adopted to denote the size of workloads dispatched to $r \in \mathcal{R}_n$ and $\overline{\chi}_{nr}$ to denote the parallelism bound when it is processed on r. It results to the constraint $0 \le x_{nr} \le \overline{\chi}_{nr}$. This constraint avoids too high degree of parallelism that leads to non-neglectable communication overhead and even unforeseen errors [34]. In addition, this formulation takes the anti-affinity, service locality, and unpredictable system failure of computing instances into consideration. When those conditions happen to resource unit r during processing the n-th job, $\overline{\chi}_{nr}$ can be set as zero online.

For the utility function for users, a zero-startup utility function is adopted, *i.e.*, $f_n : [\mathbf{0}, \overline{\boldsymbol{\chi}}_n] \to \mathbb{R}$, where $\overline{\boldsymbol{\chi}}_n := \{\overline{\chi}_{nr}\}_{r\in\mathcal{R}_n}$, as the measurement of user satisfaction for job

n. As a widely accepted assumption in previous works [35]–[37], we require $\{f_n\}_{n\in\mathcal{N}}$ to be non-decreasing, concave, and continuously differentiable on each dimension r, *e.g.*, when users receive more services, their utility should not be decreased. Proportional fairness and α-fairness are good options for $\{f_n\}_{n\in\mathcal{N}}$. It is worth noting that jobs are allowed with different utility functions in $\mathcal{N}$ because each job has its own service requirement. For each job $n \in \mathcal{N}$, its utility is a sum of separate sub-utilities achieved through each available resource unit:

$$f_n(\boldsymbol{x}_n) := \sum_{r\in\mathcal{R}_n} f_{nr}(x_{nr}), \forall n \in \mathcal{N}, \tag{2}$$

where $\boldsymbol{x}_n := \{x_{nr}\}_{r\in\mathcal{R}_n}$. For a given job n, f_{nr} can also differ on different resource units $r \in \mathcal{R}_n$. To sum up, a job can be described with the quadruple $\{\varrho_n, \mathcal{R}_n, \overline{\boldsymbol{\chi}}_n, f_n\}$.

CMEC services come with a *pay-for-value* billing model, *i.e.*, users only need to pay for the usage of the service instead of paying for the idle resources. Thus, its revenue is linearly proportional to the actual resource consumption. Formally, we define the revenue for provisioning resource $r \in \mathcal{R}$ as

$$g_{nr}(x_{nr}) := \beta_{nr} \cdot \frac{x_{nr}}{\mathcal{C}_r}, \forall n \in \mathcal{N}, r \in \mathcal{R}_n, \tag{3}$$

where $\frac{x_{nr}}{\mathcal{C}_r}$ is the fractional resource consumed by x_{nr}, and β_{nr} is a ratio indicating the unit price per resource unit for job n. For instance, β_{nr} equals to \$0.015 when the raw video's resolution is less than 1280×720 with Google Transcoder API [38]. Actually, our model is consistent with all the mainstream platform providers' pricing strategies [39]–[41].

C. Online Social Welfare Maximization

Based on the above analysis, the social welfare maximization problem can be fumulated as follows:

$$\begin{aligned}
\mathcal{P}_1 : \max_{\{\boldsymbol{x}_n\}_{n\in\mathcal{N}}} & \sum_{n\in\mathcal{N}} f_n(\boldsymbol{x}_n) + \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}_n} g_{nr}(x_{nr}) \\
s.t. & \sum_{r\in\mathcal{R}_n} x_{nr} \le \varrho_n, \forall n \in \mathcal{N}, & (4)\\
& x_{nr} = 0, \forall n \in \mathcal{N}, r \in \mathcal{R}\backslash\mathcal{R}_n, & (5)\\
& \sum_{n\in\mathcal{N}} x_{nr} \le \mathcal{C}_r, \forall r \in \mathcal{R}, & (6)\\
& 0 \le x_{nr} \le \overline{\chi}_{nr}, \forall n \in \mathcal{N}, r \in \mathcal{R}_n, & (7)
\end{aligned}$$

where (4) denotes that the sum of the workloads of all available resource units assigned to task n must be less or equal to its total workload. (5) ensures that tasks can only be dispatched to their available resource units. (6) denotes that for each resource unit r, the sum of the workloads of all tasks assigned to it must be less or equal to its maximum processing capacity. (7) ensures that each task's workload on each resource unit cannot exceed its maximum allowed parallelism.

As an online optimization problem, although $\mathcal{P}_1$ is difficult to solve[3], it is built based on complete knowledge. In online settings, the CMEC service providers should not have the information of the n-th quadruple $\{\varrho_n, \mathcal{R}_n, \overline{\boldsymbol{\chi}}_n, f_n\}$ until job n arrives. In this work, we introduce the value density to design an efficient online algorithm with the worst-case performance guarantee. It is lower and upper bounded by ι and υ, respectively:

$$\begin{cases} \iota := \min_{n\in\mathcal{N}} \min_{r\in\mathcal{R}_n} \left(\frac{\partial f_n}{\partial x_{nr}} + \frac{\beta_{nr}}{\mathcal{C}_r}\right) \\ \upsilon := \max_{n\in\mathcal{N}} \max_{r\in\mathcal{R}_n} \left(\frac{\partial f_n}{\partial x_{nr}} + \frac{\beta_{nr}}{\mathcal{C}_r}\right). \end{cases} \tag{8}$$

ι and υ denote the maximum and minimal marginal utility of job n to dispatch the workload on a resource unit r plus the unit revenue the CMEC service providers gained, respectively. They also represent the marginal social welfare in the most conservative and optimistic case of the system, *i.e.*, the minimum and maximum social welfare per unit of additional workload. ι and υ are assumed that the CEMC service supplier know them at the very beginning. Those two constants demonstrate the fluctuation of the marginal social welfare. This assumption is widely accepted in the online resource allocation problems [42]–[44]. For example, in [43], the fluctuation ratio $\frac{\upsilon}{\iota}$ is set as 36 in default. Later in the algorithm design, they help define the range of the marginal cost function.

IV. Algorithm Design with Theoretical Analysis

The key challenge to solve $\mathcal{P}_1$ in online settings is that the dispatching of each job's workloads to each resource unit is coupled because of (6), *i.e.*, each resource unit may process workload from different jobs. Nevertheless, if we could construct several feasible dual variables corresponding to $\mathcal{P}_1$, and take these dual variables as the cost for using each resource unit, a near-optimal solution could be obtained. Thus, several pseudo-social welfare functions with estimated marginal costs are constructed to do this. In this design, an important principle is utilized for solving online resource allocation problems, *i.e.*, *estimate the cost for provisioning services to each newly arrived job as a function of resource surplus*. In the following sections, we first show how pseudo-social welfare functions are designed. Then, based on these functions, `OnSocMax` is designed by solving several pseudo-social welfare maximization problems online. To guarantee `OnSocMax` is α-competitive, we demonstrate what requirements the cost functions should satisfy. Finally, we give the bound of the gap between the competitive ratio achieved by `OnSocMax` and the optimal competitive ratio of a simplified case under certain conditions.

A. Pseudo-Social Welfare Function

Due to the complexity and multiple constraints of $\mathcal{P}_1$, it is difficult to solve. Thus, we introduce its dual problem $\mathcal{P}_2$ by Lagrangian method, which is shown as follows.

Proposition 1.

$$\mathcal{P}_2 : \min_{\boldsymbol{\mu},\boldsymbol{\lambda}} \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}} \xi_{nr}(\mu_n + \lambda_r) + \sum_{n\in\mathcal{N}} \mu_n\varrho_n + \sum_{r\in\mathcal{R}} \lambda_r\mathcal{C}_r$$
$$s.t. \quad (5), (7), \boldsymbol{\mu} \ge \boldsymbol{0}, \boldsymbol{\lambda} \ge \boldsymbol{0},$$

[3]The discrete version of $\mathcal{P}_1$ is essentially a multi-dimensional 0-1 knapsack problem, which is proved to be NP-complete.

where

$$\xi_{nr}(p) := \max_{x_{nr}\in[0,\overline{\chi}_{nr}]} \Big[f_{nr}(x_{nr}) + \big(g_{nr}(x_{nr}) - p\cdot x_{nr}\big)\Big]. \quad (9)$$

Here $\boldsymbol{\mu} := \{\mu_n\}_{n\in\mathcal{N}}$ and $\boldsymbol{\lambda} := \{\lambda_r\}_{r\in\mathcal{R}}$ are the dual variables corresponding to (4) *and* (6)*, respectively. $\xi_{nr}(\cdot)$ is known as the convex conjugate of the fractional social welfare $f_{nr}+g_{nr}$. It is shown that p can denote the marginal cost for providing unit workload of CMEC service providers. It can also be expressed as:*

$$\xi_{nr}(p) := \max_{x_{nr}\in[0,\overline{\chi}_{nr}]} \Big[\big(f_{nr}(x_{nr}) - p\cdot x_{nr}\big) + g_{nr}(x_{nr})\Big], \quad (10)$$

where p can denote the marginal cost for processing unit workload by users.

Proof. The result is obtained with Lagrangian. After introducing the Lagrange multipliers μ and λ, we can construct the Lagrange function as:

$$L(x,\mu,\lambda) = \sum_{n\in N} f_n(x_n) + \sum_{n\in N}\sum_{r\in R_n} g_{nr}(x_{nr}) - \sum_{n\in N}\mu_n\left(\sum_{r\in R_n} x_{nr} - \varrho_n\right) - \sum_{r\in R}\lambda_r\left(\sum_{n\in N} x_{nr} - C_r\right). \quad (11)$$

Then, $\mathcal{P}_1$ is transformed into a minimization problem $\mathcal{P}_2$ through the duality method. It aims to minimize the expected value of the Lagrangian function for the dual variables μ and λ, *i.e.,* $\mathcal{P}_2$. Then, a maximization operation is performed for each x_{nr} to find the optimal resource allocation strategy. Specifically, we aim to maximize the net benefit of task n on resource unit r when given the price $p = \mu_n + \lambda_r$, which can be achieved through the maximum of each job's local optimization problem $\xi_{nr}(p)$. □

Thus, for each arrived job n, we define the pseudo-social welfare function, denoted by $\tilde{\mathcal{W}}_n(\boldsymbol{x}_n)$, as

$$\Big(f_n(\boldsymbol{x}_n) - \sum_{r\in\mathcal{R}_n}\int_{\omega_r^{(n)}}^{\omega_r^{(n)}+x_{nr}} \phi_r(u)du\Big) + \sum_{r\in\mathcal{R}_n} g_{nr}(x_{nr}), \quad (12)$$

where $\omega_r^{(n)}$ denotes the resource usage level of job n on resources unit r. ϕ_r is a non-decreasing estimation of the marginal cost for the resource unit $r\in\mathcal{R}$ processing unit workload when the resource surplus $u\in[0,\mathcal{C}_r]$. We also define $\phi_r(u) = +\infty$ when $u > \mathcal{C}_r$. The non-decreasing property profoundly reflects an underlying economic phenomenon, *i.e.*, a thing is valued in proportion to its rarity. The later a job arrives, the higher cost it has to pay. The first component is the pseudo-utility of job n, which is the utility of it minus the cost to pay. The second component is the platform's revenue. This is corresponding to (10). In addition, if we organize $\tilde{\mathcal{W}}_n(\boldsymbol{x}_n)$ as

$$f_n(\boldsymbol{x}_n) + \sum_{r\in\mathcal{R}_n}\Big(g_{nr}(x_{nr}) - \int_{\omega_r^{(n)}}^{\omega_r^{(n)}+x_{nr}} \phi_r(u)du\Big), \quad (13)$$

it is corresponding to (9). The second component can be regarded as the *net profit* of the platform for serving job n. In this case, the later a job arrives, the harder the resource surplus to meet its requirements, which results in higher costs. The following content applies to both of these two interpretations.

In that case, the designed pseudo welfare function $\tilde{\mathcal{W}}_n$ has a very close relationship to $\mathcal{P}_2$. To bridge connections between the optimal dual variables of $\mathcal{P}_1$ and the optimal solution $\boldsymbol{x}_n^\star$ that maximizes $\tilde{\mathcal{W}}_n$, if we could find appropriate $p^\star$ and $\boldsymbol{x}_n^\star$, we can bridge their connection through

$$\tilde{\mathcal{W}}_n(\boldsymbol{x}_n^\star) \approx \sum_{r\in\mathcal{R}_n}\xi_{nr}(p^\star). \quad (14)$$

Based on this, we can interpret p as the marginal cost for processing unit workload. We bridge the subtle connection between ξ_{nr} and $\tilde{\mathcal{W}}_n$ in the following proposition, which is crucial for the design of `OnSocMax`.

Proposition 2. *$\forall n\in\mathcal{N}, r\in\mathcal{R}$, when $\phi_r(\mathcal{C}_r)\ge\upsilon$, if (i) $\boldsymbol{x}_n^\star = \{x_{nr}^\star\}_{r\in\mathcal{R}_n}$ and $\mu_n^\star$ are respectively the optimal primal and dual solutions to* (4) *of the following problem $\mathcal{P}_3$:*

$$\mathcal{P}_3 : \max_{\boldsymbol{x}_n}\tilde{\mathcal{W}}_n(\boldsymbol{x}_n) \quad s.t. \quad (4),(5),(7),$$

and (ii) the resource usage level ω_r is updated by the optimal solution of each job, i.e.,

$$\begin{cases}\omega_r^{(n+1)} = \omega_r^{(n)} + x_{nr}^\star\\ \omega_r^{(1)} = 0,\end{cases} \quad (15)$$

then, $x_{nr}^\star$ is also the optimal solution that maximizes $\xi_{nr}(p)$ given $p = \phi_r(\omega_r^{(n+1)}) + \mu_n^\star$, where $\phi_r(\omega_r^{(n+1)})$ is used to estimate $\lambda_r^\star$. Thus, we need to prove the following equation holds:

$$\xi_{nr}\Big(\phi_r(\omega_r^{(n+1)}) + \mu_n^\star\Big) = f_{nr}(x_{nr}^\star) + g_{nr}(x_{nr}^\star) - \Big(\phi_r(\omega_r^{(n+1)}) + \mu_n^\star\Big)x_{nr}^\star. \quad (16)$$

Proof. By the definition of the non-decreasing marginal cost function $\phi_r(\cdot)$, we can find that it is discontinuous at $\mathcal{C}_r$. Thus, when $\phi_r(\mathcal{C}_r)\ge\upsilon$, there must exist a resource usage level $\overline{\omega}_r\le\mathcal{C}_r$ such that $\phi_r(\overline{\omega}_r)=\upsilon$. Note that the function $f_n + \sum_{r\in\mathcal{R}_n} g_{nr}$ is non-decreasing and its derivative on r is not more than υ. Therefore, when the input of ϕ_r is $\omega_r^{(n)} + x_{nr}$, suppose $\omega_r^{(n)} + x_{nr}\le\overline{\omega}_r$. Consequently, the derivative of the integral function

$$\Phi_r(x_{nr}) := \int_{\omega_r^{(n)}}^{\omega_r^{(n)}+x_{nr}}\phi_r(u)du, \quad (17)$$

is continuous, non-decreasing, and convex when $x_{nr}\le\overline{\omega}_r - \omega_r^{(n)}$. The convexity is because Φ_r', *i.e.,* ϕ_r, is non-decreasing and its derivative is greater than zero. Thus, $\mathcal{P}_3$ is a convex optimization program, and its optimal solution can be obtained through KKT conditions. Let $x_{nr}^\star$ denotes the optimal primal solution of $\mathcal{P}_3$. $\mu_n^\star$, $\gamma_{nr}^\star$, and $\zeta_{nr}^\star$ denote the optimal dual solutions of $\mathcal{P}_3$ ($\mu_n^\star$ to (4) while γ_{nr} and ζ_{nr} to the right part and left part of (7), respectively). The Karush-Kuhn-Tucker (KKT) conditions of $\mathcal{P}_3$ are listed below:

$$\begin{cases} f'_{nr}(x^\star_{nr}) + \frac{\beta_{nr}}{\mathcal{C}_r} - \phi_r\big(\omega_r^{(n+1)}\big) = \mu^\star_n + \gamma^\star_{nr} - \zeta^\star_{nr} \\ \mu^\star_n \Big(\sum_{r\in\mathcal{R}_n} x^\star_{nr} - \varrho_n\Big) = 0 \\ \zeta^\star_{nr} \cdot x^\star_{nr} = 0 \\ \gamma^\star_{nr}\big(x^\star_{nr} - \overline{\chi}_{nr}\big) = 0. \end{cases} \tag{18}$$

The first equation of (18) denotes the gradient of $\mathcal{P}_3$ at x_{nr} direction is zero, and the remaining three equations represent the complementary slackness condition of (4) and (7). With KKT conditions (18), we show that the optimal solution $x^\star_{nr}$ of $\mathcal{P}_3$ simultaneously optimizes the conjugate $\xi_{nr}(p)$ given $p = \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$. Specifically, three different cases are discussed.

- **Case I**: When $f'_{nr}(x^\star_{nr}) + \frac{\beta_{nr}}{\mathcal{C}_r} > \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$, $\tilde{\mathcal{W}}_n$ is an increasing function on dimension r under (4). Thus, we have $x^\star_{nr} = \overline{\chi}_{nr}$, which leads to
$$f'_{nr}(\overline{\chi}_{nr}) + \frac{\beta_{nr}}{\mathcal{C}_r} > \phi_r(\omega_r^{(n+1)}) + \mu^\star_n. \tag{19}$$
(19) is the derivative of $f_{nr}(x_{nr}) + \big(g_{nr}(x_{nr}) - px_{nr}\big)$ by setting p as $\phi_r(\omega_r^{(n+1)})+\mu^\star_n$. Thus, it is monotone increasing in the feasible region $[0, \overline{\chi}_{nr}]$. Therefore, $x^\star_{nr} = \overline{\chi}_{nr} = \operatorname{argmax}_{0\le x_{nr}\le\overline{\chi}_{nr}} \big[f_{nr}(x_{nr})+g_{nr}(x_{nr})-p\cdot x_{nr}\big]$, which means the same $x^\star_{nr}$ maximizes both $\mathcal{P}_3$ and the conjugate simultaneously given $p = \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$. Thus, (16) holds.
- **Case II**: When $f'_{nr}(x^\star_{nr}) + \frac{\beta_{nr}}{\mathcal{C}_r} < \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$, similarly, we have $x^\star_{nr} = 0$, which leads to
$$f'_{nr}(0) + \frac{\beta_{nr}}{\mathcal{C}_r} < \phi_r(\omega_r^{(n)}) + \mu^\star_n, \tag{20}$$
and $\omega_r^{(n+1)} = \omega_r^{(n)} + 0 = \omega_r^{(n)}$. Analogously, (20) means that $f_{nr}(x_{nr})+\big(g_{nr}(x_{nr})-p\cdot x_{nr}\big)$ is monotone decreasing in feasible region $[0, \overline{\chi}_{nr}]$ given $p = \phi_r(\omega_r^{(n+1)})+\mu^\star_n$. Therefore, $x^\star_{nr} = 0 = \operatorname{argmax}_{0\le x_{nr}\le\overline{\chi}_{nr}} \big[f_{nr}(x_{nr}) + g_{nr}(x_{nr}) - p\cdot x_{nr}\big]$, which also leads to (16).
- **Case III**: When $f'_{nr}(x^\star_{nr}) + \frac{\beta_{nr}}{\mathcal{C}_r} = \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$, $x^\star_{nr}$ is an maximum of $f_{nr}(x_{nr}) + \big(g_{nr}(x_{nr}) - p\cdot x_{nr}\big)$ given $p = \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$. In addition, $\zeta^\star_{nr}\cdot x^\star_{nr} = 0$, $\gamma^\star_{nr}\big(x^\star_{nr} - \overline{\chi}_{nr}\big) = 0$, $x^\star_{nr} \ne 0$, and $x^\star_{nr} \ne \overline{\chi}_{nr}$. Thus, $\gamma^\star_{nr} = \zeta^\star_{nr} = 0$. $x^\star_{nr} \in \operatorname{argmax}_{0\le x_{nr}\le\overline{\chi}_{nr}} \big[f_{nr}(x_{nr}) + g_{nr}(x_{nr}) - p\cdot x_{nr}\big]$, which means (16) holds.

In summary, all three conditions prove that (16) holds. They are visualized in Fig. 3. □

So far, we have analyzed the properties of the pseudo-social welfare functions and the conjugates. We prove that the optimal solution $x^\star_{nr}$ of the pseudo-social welfare functions is also the optimal solution that maximizes the conjugates $\xi_{nr}(p)$ given $p = \phi_r(\omega_r^{(n+1)}) + \mu^\star_n$. In addition, Fig. 4 shows the relation of the proposed $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ and the conjugate. In the following sections, we will first give the design details of the online algorithm `OnSocMax` by solving $\mathcal{P}_3$. Then, we illustrate what requirements the marginal cost functions $\{\phi_r\}_{r\in\mathcal{R}}$ should satisfy to make `OnSocMax` α-competitive for some underlying α.

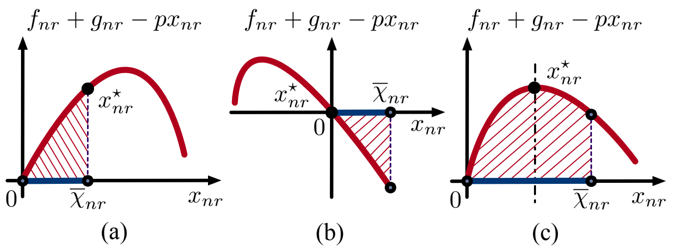


Fig. 3. A visualization on how the three conditions affect the optimal $x^\star_{nr}$ of $f_{nr}(x_{nr}) + g_{nr}(x_{nr}) - \big(\phi_r(\omega_r^{(n+1)}) + \mu^\star_n\big)\cdot x_{nr}$, respectively.

B. *`OnSocMax` Design and Analysis*

`OnSocMax` is an online workload scheduling framework built on solving $\mathcal{P}_3$ for each newly arrived job n in sequence. A hat is placed on top of variables that denote the variables involved in the framework. The procedure of `OnSocMax` is captured in **Algorithm 1**. Specifically, upon the arrival of each request in sequence, `OnSocMax` solves $\mathcal{P}_3$ and updates resource usage level ω_r after receiving the job information. This ensures that an optimal service request dispatching and workload scheduling scheme can be quickly identified when each request arrives. The three conditions (a), (b), and (c) in Step 6 correspond to the three conditions in Fig. 3, respectively.

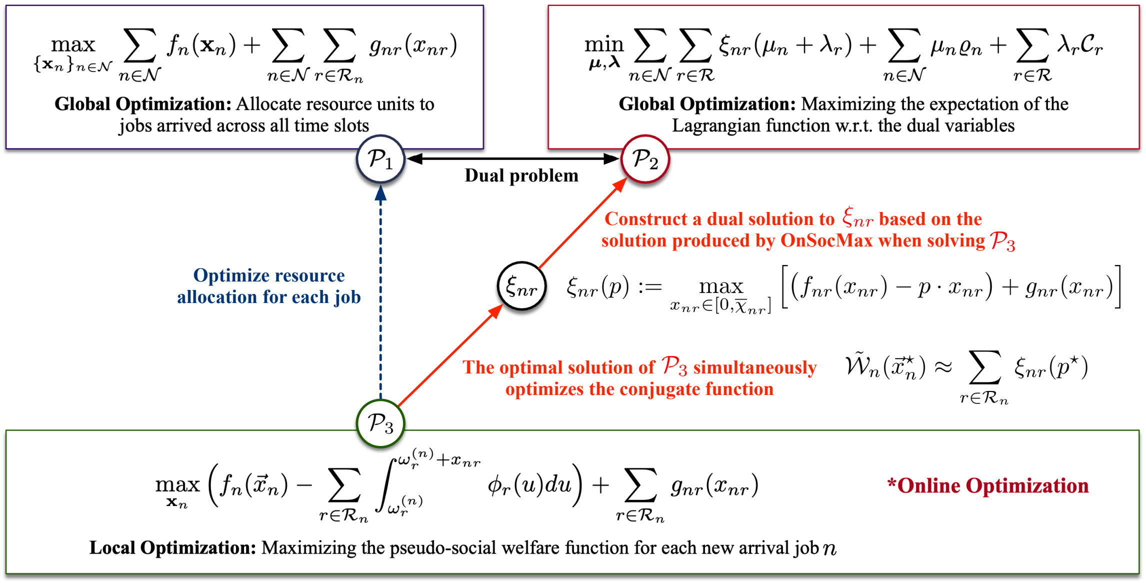
`OnSocMax` is at most polynomial because $\mathcal{P}_3$ is convex. Several methods can solve this problem, *e.g.*, intelligent optimization algorithm [45] and augmented Lagrangian method (ALM). However, intelligent optimization algorithms lack the theoretical guarantee of finding a globally optimal solution and waste a large amount of computational resources in the fitness evaluation. Since the objective function $\mathcal{P}_3$ is convex and the constraints are linear, ALM is guaranteed to find the globally optimal solution by using the KKT condition. In addition, $\{\hat{\boldsymbol{x}}_n\}_{n\in\mathcal{N}}$ obtained by `OnSocMax` is feasible to $\mathcal{P}_1$. To quantify how "good" `OnSocMax` is, the standard competitive analysis framework is adopted. The definition of competitive ratio is shown as follows.

Definition 1. *For any arrival instance $\mathcal{A}$ of all jobs $n\in\mathcal{N}$, the competitive ratio for an online algorithm is defined as*

$$\alpha := \max_{\forall\mathcal{A}} \frac{\Theta^\star_{\mathcal{P}_1}(\mathcal{A})}{\Theta_{on}(\mathcal{A})}, \tag{21}$$

where $\Theta^\star_{\mathcal{P}_1}(\mathcal{A})$ is the maximum objective value of $\mathcal{P}_1$, $\Theta_{on}(\mathcal{A})$ is the objective function value of $\mathcal{P}_1$ obtained by this online algorithm.

The competitive ratio quantifies the worst-case ratio between the optimum and the objective obtained by the online algorithm. In addition, the smaller α indicates the online algorithm is better. An online algorithm is called α-*competitive* if its ratio is upper bounded by α. To guarantee that `OnSocMax` is α-competitive, the requirements of the marginal cost functions $\{\hat{\phi}_r\}_{r\in\mathcal{R}}$ are given for some α. Most importantly, this work gives the detailed formulation of $\hat{\phi}_r(\omega)$, which is irrelevant with the utilities $\{f_n\}_{n\in\mathcal{N}}$ and $\{g_{nr}\}_{n\in\mathcal{N},r\in\mathcal{R}}$, and only corresponding to ω, υ, ι, α, and $\mathcal{C}_r$. The marginal cost



$\max_{\{\mathbf{x}_n\}_{n\in\mathcal{N}}} \sum_{n\in\mathcal{N}} f_n(\mathbf{x}_n) + \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}_n} g_{nr}(x_{nr})$
Global Optimization: Allocate resource units to jobs arrived across all time slots
$\min_{\mu,\lambda} \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}} \xi_{nr}(\mu_n+\lambda_r) + \sum_{n\in\mathcal{N}} \mu_n\varrho_n + \sum_{r\in\mathcal{R}} \lambda_r\mathcal{C}_r$
Global Optimization: Maximizing the expectation of the Lagrangian function w.r.t. the dual variables
$\mathcal{P}_1$
Dual problem
$\mathcal{P}_2$
Construct a dual solution to ξ_{nr} based on the solution produced by OnSocMax when solving $\mathcal{P}_3$
Optimize resource allocation for each job
ξ_{nr}
$\xi_{nr}(p) := \max_{x_{nr}\in[0,\overline{\chi}_{nr}]} \left[(f_{nr}(x_{nr}) - p\cdot x_{nr}) + g_{nr}(x_{nr})\right]$
The optimal solution of $\mathcal{P}_3$ simultaneously optimizes the conjugate function
$\widetilde{W}_n(\vec{x}_n^\star) \approx \sum_{r\in\mathcal{R}_n} \xi_{nr}(p^\star)$
$\mathcal{P}_3$
$\max_{\mathbf{x}_n} \left(f_n(\vec{x}_n) - \sum_{r\in\mathcal{R}_n}\int_{\omega_r^{(n)}}^{\omega_r^{(n)}+x_{nr}} \phi_r(u)du\right) + \sum_{r\in\mathcal{R}_n} g_{nr}(x_{nr})$
*Online Optimization
Local Optimization: Maximizing the pseudo-social welfare function for each new arrival job n

Fig. 4. Relation of $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, and the conjugate. OnSocMax is designed by solving $\mathcal{P}_3$ online.

Algorithm 1: OnSocMax

```
Input: $\{\mathcal{C}_r\}_{r\in\mathcal{R}}$ and $\{g_{nr}\}_{n\in\mathcal{N},r\in\mathcal{R}}$
Output: Online solution to $\mathcal{P}_1$ and final utilizations
        for the resource mesh
1  $\forall r\in\mathcal{R} : \hat{\omega}_r^{(1)} \leftarrow 0$
2  while a new job $n$ arrives do
3      Receive the quadruple $\{\varrho_n, \mathcal{R}_n, \overline{\chi}_n, f_n\}$
4      /* Solve the convex program $\mathcal{P}_3$ optimally */
5      for $r\in\mathcal{R}_n$ do
6          Get $\hat{x}_{nr}$ with KKT conditions (18) by
               $\hat{x}_{nr} = \begin{cases} \overline{\chi}_{nr} & (a)\\ 0 & (b)\\ (f'_{nr})^{-1}(\hat{\phi}_r(\hat{\omega}_r^{(n+1)}) + \hat{\mu}_n - \frac{\beta_{nr}}{\mathcal{C}_r}) & (c)\end{cases}$
7          $\hat{\omega}_r^{(n+1)} \leftarrow \hat{\omega}_r^{(n)} + \hat{x}_{nr}$ // Update utilization
8      end for
9      $n \leftarrow n+1$
10 end while
11 return $\{\hat{\boldsymbol{x}}_n\}_{n\in\mathcal{N}}$ and $\{\hat{\omega}_r^{(|\mathcal{N}|+1)}\}_{r\in\mathcal{R}}$
```

function $\hat{\phi}_r$ used in step 6-condition (c) of OnSocMax is designed as

$$\hat{\phi}_r(\omega) = \begin{cases} \iota & \omega\in[0,\hat{\varpi}_r)\\ \frac{\upsilon-\iota}{\exp(\hat{\alpha})-\exp(\frac{\hat{\alpha}}{\hat{\alpha}-1})} e^{\left(\frac{\hat{\alpha}}{\mathcal{C}_r}\omega\right)} + \frac{\iota}{\hat{\alpha}} & \omega\in[\hat{\varpi}_r,\mathcal{C}_r]\\ +\infty & \omega\in(\mathcal{C}_r,+\infty), \end{cases} \tag{22}$$

where $\hat{\varpi}_r$ is a resource utilization threshold and $\hat{\alpha}$ can be obtained in the following section. To prove this result, let $\frac{\upsilon-\iota}{\exp(\hat{\alpha})-\exp(\frac{\hat{\alpha}}{\hat{\alpha}-1})} e^{\left(\frac{\hat{\alpha}}{\mathcal{C}_r}\omega\right)} + \frac{\iota}{\hat{\alpha}} = \hat{\varphi}_r(\omega)$ first.

Theorem 1. *(Extended from Theorem 4 of [46]) OnSocMax is α-competitive for some $\alpha \ge 1$ if $\forall r\in\mathcal{R}$, the marginal cost function $\hat{\phi}_r$ is divided into three segments, including flat, increasing, and infinite segments. It is in the form of*

$$\hat{\phi}_r(\omega) = \begin{cases} \iota & \omega\in[0,\hat{\varpi}_r)\\ \hat{\varphi}_r(\omega) & \omega\in[\hat{\varpi}_r,\mathcal{C}_r]\\ +\infty & \omega\in(\mathcal{C}_r,+\infty), \end{cases} \tag{23}$$

where the minimum value density across all jobs is bounded by ι. Therefore, service requests are guaranteed to be fulfilled whenever resource utilization remains under $\hat{\varpi}_r$, irrespective of the associated values. Based on [46], $\hat{\varphi}_r$ is a non-decreasing function that satisfies

$$\begin{cases} \hat{\varphi}_r(\omega)\mathcal{C}_r \le \alpha\int_0^\omega \hat{\phi}_r(u)du - \iota\cdot\omega, & \omega\in[\hat{\varpi}_r,\mathcal{C}_r]\\ \hat{\varphi}_r(\hat{\varpi}_r) = \iota, \hat{\varphi}_r(\mathcal{C}_r)\ge\upsilon. \end{cases} \tag{24}$$

Proposition 3. *$\mathcal{B} := \{\mathcal{A}_1, \mathcal{A}_2, ...\}$ denotes the set of arrival instances of all jobs, and $\Theta_{\mathcal{P}_2}(\mathcal{A})$ denotes a feasible objective value of the dual problem $\mathcal{P}_2$ for any arrival instance $\mathcal{A}$. Hereinafter, $\hat{\omega}_r^{(|\mathcal{N}|+1)}$ is replaced to $\hat{\omega}_r^N$ for simplification. Furthermore, $\mathcal{B}$ is divided into three disjoint sets:*

$$\begin{cases} \mathcal{B}_1 := \{\mathcal{A} \mid 0\le\hat{\omega}_r^N < \hat{\varpi}_r, \forall r\in\mathcal{R}\}\\ \mathcal{B}_2 := \{\mathcal{A} \mid \hat{\varpi}_r \le \hat{\omega}_r^N \le \mathcal{C}_r, \forall r\in\mathcal{R}\}\\ \mathcal{B}_3 := \mathcal{B}\backslash(\mathcal{B}_1\cup\mathcal{B}_2). \end{cases} \tag{25}$$

$\mathcal{B}_1$ and $\mathcal{B}_2$ contain the instances whose final utilizations of all resource units in the pool are below and above the threshold $\hat{\varpi}_r$, respectively. The goal is to prove that, under the conditions (23) and (24), $\forall\mathcal{A}\in\mathcal{B}_1,\mathcal{B}_2,\mathcal{B}_3$ respectively, the following relations hold:

$$\alpha\cdot\Theta_{on}(\mathcal{A}) \ge \Theta_{\mathcal{P}_2}(\mathcal{A}) \ge \Theta^\star_{\mathcal{P}_1}(\mathcal{A}). \tag{26}$$

In the following analysis, we drop the parentheses and $\mathcal{A}$ for simplification.

Proof. A technique named *instance-dependent online primal-dual approach* is adopted to prove this. The key idea is to construct a dual solution to $\mathcal{P}_2$ (μ_n and λ_r) based on the solution $\{\hat{\boldsymbol{x}}_n\}_{n\in\mathcal{N}}$ produced by OnSocMax. Specifically, μ

is constructed through KKT conditions, and λ is estimated by marginal cost. Then, it uses this dual objective to build the upper bound of the optimum of $\mathcal{P}_1$ based on weak duality. When building the upper bound, this technique studies *the worst-case instances* under different scenarios.

Case I: $\forall \mathcal{A} \in \mathcal{B}_1$, the marginal costs experienced by all jobs are the same according to (23), *i.e.*, ι. In this case, each job n is processed with the maximum parallelism rate $\overline{\chi}_{nr}$ on $r \in \mathcal{R}_n$. Thus, $\Theta^\star_{\mathcal{P}_1}/\Theta_{on} = 1 \leq \alpha$.

Case II: $\forall \mathcal{A} \in \mathcal{B}_2$, we construct a feasible dual solution to $\mathcal{P}_2$ as

$$\begin{cases} \hat{\mu}_n = \mu^\star_n, & \forall n \in \mathcal{N} \\ \hat{\lambda}_r = \hat{\phi}_r(\hat{\omega}^N_r) & \forall r \in \mathcal{R}, \end{cases} \tag{27}$$

where $\mu^\star_n$ is the optimal dual solution to $\mathcal{P}_3$ introduced by (18). Let $p \geq p' \geq 0$ and denote the optimal solution that maximizes the conjugate $\xi_{nr}(p)$ by $\tilde{x}_{nr}$ given p. Then,

$$\begin{aligned} \xi_{nr}(p) &= f_{nr}(\tilde{x}_{nr}) + \big(g_{nr}(\tilde{x}_{nr}) - p \cdot \tilde{x}_{nr}\big) \\ &\leq f_{nr}(\tilde{x}_{nr}) + \big(g_{nr}(\tilde{x}_{nr}) - p' \cdot \tilde{x}_{nr}\big) \\ &= \max_{x_{nr}} \Big[f_{nr}(x_{nr}) + \big(g_{nr}(x_{nr}) - p' \cdot x_{nr}\big)\Big] \\ &= \xi_{nr}(p'), \end{aligned} \tag{28}$$

which indicates that the conjugate $\xi_{nr}(p)$ is non-increasing with p. The above derivation uses the fact that $f_{nr} + g_{nr}$ is non-decreasing. Based on weak duality and the non-increasing property of the conjugate, we have

$$\begin{aligned} \Theta^\star_{\mathcal{P}_1} &\leq \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \xi_{nr}\Big(\mu^\star_n + \hat{\phi}_r\big(\hat{\omega}^N_r\big)\Big) + \sum_{n\in\mathcal{N}} \mu^\star_n \varrho_n \\ &\quad + \sum_{r\in\mathcal{R}} \hat{\phi}_r\big(\hat{\omega}^N_r\big)\mathcal{C}_r \qquad \triangleright \textit{the right-side is } \Theta_{\mathcal{P}_2} \\ &\leq \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \xi_{nr}\Big(\mu^\star_n + \hat{\phi}_r\big(\hat{\omega}^{(n+1)}_r\big)\Big) + \sum_{n\in\mathcal{N}} \mu^\star_n \varrho_n \\ &\quad + \sum_{r\in\mathcal{R}} \hat{\phi}_r\big(\hat{\omega}^N_r\big)\mathcal{C}_r \qquad \triangleright (28) \\ &= \sum_{r\in\mathcal{R}} \Big(\hat{\phi}_r\big(\hat{\omega}^N_r\big)\mathcal{C}_r - \sum_{n\in\mathcal{N}} \hat{\phi}_r\big(\hat{\omega}^{(n+1)}_r\big)\hat{x}_{nr}\Big) \\ &\quad + \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big) := \Theta_{tmp}. \triangleright (16) \end{aligned}$$

The first equation is built on the weak duality of $\mathcal{P}_1$ and $\mathcal{P}_2$. The last equality holds because $\hat{x}_{nr}$ simultaneously maximizes $\mathcal{P}_3$ and the conjugate $\xi_{nr}\big(\mu^\star_n + \hat{\phi}_r(\hat{\omega}^N_r)\big)$ (result of **Proposition 2**). Since $\{\hat{\phi}_r\}_{r\in\mathcal{R}}$ are non-decreasing, $\forall n \in \mathcal{N}, r \in \mathcal{R}$,

$$\hat{\phi}_r\big(\hat{\omega}^{(n+1)}_r\big)\hat{x}_{nr} \geq \int_{\hat{\omega}^{(n)}_r}^{\hat{\omega}^{(n+1)}_r} \hat{\phi}_r(u)du. \tag{29}$$

(29) is illustrated in Fig. 5. It is worth noting that $\{\hat{\phi}_r\}_{r\in\mathcal{R}}$ is a convex upward function, which is shown in (36) in the following part. Further, we have

$$\sum_{n\in\mathcal{N}} \hat{\phi}_r\big(\hat{\omega}^{(n+1)}_r\big)\hat{x}_{nr} \geq \int_{\hat{\omega}^{(1)}_r}^{\hat{\omega}^N_r} \hat{\phi}_r(u)du, \tag{30}$$

where $\hat{\omega}^{(1)}_r = 0$ because of (15). It shows that the sum of the marginal costs assigned to jobs is always greater or equal to the cumulative marginal cost. Besides, from Fig. 3 we can find that $\xi_{nr}(\mu^\star_n + \hat{\phi}_r(\hat{\omega}^{(n+1)}_r)) \geq 0$ holds for all jobs. Thus, based on (30), we have

$$\sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big) \geq \int_0^{\hat{\omega}^N_r} \hat{\phi}_r(u)du. \tag{31}$$

The right side of the inequality drops $\mu^\star_n \hat{x}_{nr}$. Based on the above results (30) and (31), we have

$$\begin{aligned} \Theta_{tmp} &\leq \sum_{r\in\mathcal{R}} \Big(\hat{\phi}_r\big(\hat{\omega}^N_r\big)\mathcal{C}_r - \int_0^{\hat{\omega}^N_r} \hat{\phi}_r(u)du\Big) \\ &\quad + \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big) \qquad \triangleright (30) \\ &< \sum_{r\in\mathcal{R}} (\alpha - 1) \int_0^{\hat{\omega}^N_r} \hat{\phi}_r(u)du \qquad \triangleright (24)\ \&\ \textit{drop}\ \iota \cdot \hat{\omega}^N_r \\ &\quad + \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big) \\ &\leq \sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big)\alpha. \qquad \triangleright (31) \end{aligned}$$

The final expression is exactly $\alpha \cdot \Theta_{on}$. Thus, $\Theta^\star_{\mathcal{P}_1}/\Theta_{on} < \alpha$.

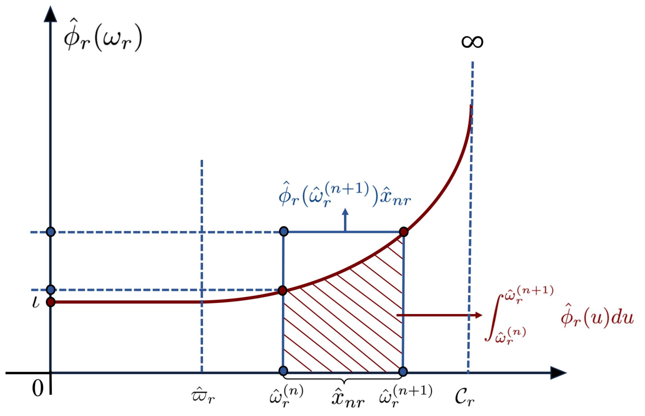


Fig. 5. A visualization of the inequality (29). The area of the blue rectangle is not less than the area of the shaded region because of the non-decreasing property of $\hat{\phi}_r$.

Case III: $\forall \mathcal{A} \in \mathcal{B}_3$, we define two disjoint sets to split the resource pool $\mathcal{R}$:

$$\begin{cases} \mathcal{R}_1 := \{r \in \mathcal{R} \mid 0 \leq \hat{\omega}^N_r < \hat{\varpi}_r, \forall r \in \mathcal{R}\} \\ \mathcal{R}_2 := \{r \in \mathcal{R} \mid \hat{\varpi}_r \leq \hat{\omega}^N_r \leq \mathcal{C}_r, \forall r \in \mathcal{R}\}. \end{cases} \tag{32}$$

For resource unit r in different sets, the corresponding dual variables are constructed in different ways. For resource units in $\mathcal{R}_1$, their usage should not exceed the total amount actually utilized throughout the entire dispatching process. We extend $\mathcal{P}_1$ to $\mathcal{P}'_1$ by adding the following constraint:

$$\sum_{n\in\mathcal{N}} \sum_{r\in\mathcal{R}_1} x_{nr} \leq \sum_{r\in\mathcal{R}} \hat{\omega}^N_r. \tag{33}$$

Apparently, $\mathcal{P}_1'$ is the same as $\mathcal{P}_1$ for `OnSocMax` since (33) is not violated by $\{\hat{\boldsymbol{x}}_n\}_{n\in\mathcal{N}}$. The dual problem $\mathcal{P}_2'$ to $\mathcal{P}_1'$ is

$$\begin{aligned}\mathcal{P}_2': \min_{\boldsymbol{\mu},\boldsymbol{\lambda}} \sum_{n\in\mathcal{N}} \Big[&\sum_{r\in\mathcal{R}_1} \xi_{nr}(\mu_n+\lambda_r+\delta) + \sum_{r\in\mathcal{R}_2} \xi_{nr}(\mu_n+\lambda_r)\Big] \\ &+ \sum_{n\in\mathcal{N}} \mu_n \varrho_n + \sum_{r\in\mathcal{R}} \lambda_r \mathcal{C}_r + \delta \sum_{r\in\mathcal{R}} \hat{\omega}_r^N \\ &s.t. \quad (5),(7), \boldsymbol{\mu}\geq \mathbf{0}, \boldsymbol{\lambda}\geq\mathbf{0}, \delta\geq 0,\end{aligned}$$

where δ is the dual variable corresponding to the newly added constraint (33). Then, we construct the dual solution to $\mathcal{P}_2'$ as

$$\begin{cases} \hat{\lambda}_r = \begin{cases} 0 & \forall r\in\mathcal{R}_1 \\ \hat{\phi}_r(\omega_r^N) & \forall r\in\mathcal{R}_2 \end{cases} \\ \delta = \iota \\ \hat{\mu}_n = \mu_n^\star \quad \forall n\in\mathcal{N}. \end{cases} \tag{35}$$

Based on (35), we can follow a similar approach as show in **Case II** to obtain that $\Theta^\star_{\mathcal{P}_1}/\Theta_{on} \leq \alpha$. A slight difference is that, in **Case III**, when applying (24) to Θ'_{tmp}, the result is tightly bounded, *i.e.*,

$$\begin{aligned}\Theta^\star_{\mathcal{P}_1} &\leq \sum_{r\in\mathcal{R}_2} \Big(\hat{\phi}_r(\hat{\omega}_r^N)\mathcal{C}_r - \int_0^{\hat{\omega}_r^N} \hat{\phi}_r(u)du \Big) \\ &\quad + \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big) + \iota \sum_{r\in\mathcal{R}} \hat{\omega}_r^N \;\triangleright (30) \\ &< \sum_{r\in\mathcal{R}_2} \big(\alpha-1\big) \int_0^{\hat{\omega}_r^N} \hat{\phi}_r(u)du \quad \triangleright (24) \\ &\quad + \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big) \\ &\leq \sum_{n\in\mathcal{N}}\sum_{r\in\mathcal{R}} \Big(f_{nr}(\hat{x}_{nr}) + g_{nr}(\hat{x}_{nr})\Big)\alpha. \quad \triangleright (31)\end{aligned}$$

□

Theorem 1 extends the Two-point Boundary Value ODEs for designing the marginal cost functions from standard 0-1 knapsack problem to multi-dimensional fractional problems. Based on **Theorem 1** and Gronwall's Inequality, we have the detailed design of $\{\hat{\phi}_r\}_{r\in\mathcal{R}}$, which is irrelevant with the utilities $\{f_n\}_{n\in\mathcal{N}}$ and $\{g_{nr}\}_{n\in\mathcal{N},r\in\mathcal{R}}$, as follows.

Theorem 2. *$\forall r\in\mathcal{R}_n$, if the marginal cost function $\hat{\phi}_r$ used in step 6-condition (c) of `OnSocMax` is designed as (22). Moreover, it is a convex upward function when $\omega\in[\hat{\varpi}_r, \mathcal{C}_r]$ because its first and second-order derivatives are greater than zero, i.e.,*

$$\begin{cases} \hat{\phi}_r'(\omega) = \frac{\upsilon-\iota}{e^{\alpha}-e^{\alpha-1}} \cdot \frac{\alpha}{C_r} e^{\left(\frac{\alpha}{C_r}\omega\right)} > 0 \\ \hat{\phi}_r''(\omega) = \frac{\upsilon-\iota}{e^{\alpha}-e^{\alpha-1}} \cdot \left(\frac{\alpha}{C_r}\right)^2 e^{\left(\frac{\alpha}{C_r}\omega\right)} > 0. \end{cases} \tag{36}$$

In (22), $\hat{\varpi}_r = \frac{\mathcal{C}_r}{\hat{\alpha}-1}$, then (i) `OnSocMax` is $\hat{\alpha}$-competitive, where $\hat{\alpha}$ is the solution of

$$\hat{\alpha} = \frac{\hat{\alpha}}{\hat{\alpha}-1} + \ln \frac{\hat{\alpha}\frac{\upsilon}{\iota}-1}{\hat{\alpha}-1}, \tag{37}$$

and (ii) when $\hat{\alpha}\geq\frac{\iota}{\upsilon}+1$, the gap between $\hat{\alpha}$ and the optimal competitive ratio when $|\mathcal{R}|=1$ is at least $\frac{2}{\sqrt{5}+1} - \ln\frac{\sqrt{5}+1}{2} \approx 0.1368$.

Proof. We first introduce Gronwall's inequality as follows. *$\forall x\in[\underline{x},\overline{x}]$, if $f(x)\leq a(x)+b(x)\int_{\underline{x}}^x f(u)du$, then*

$$f(x) \leq a(x) + b(x)\int_{\underline{x}}^x a(u)\Big(\int_u^x b(w)dw\Big)du, \tag{38}$$

where $f(x)$ is continuous, $a(x)$ and $b(x)$ are integrable and $\forall x\in[\underline{x},\overline{x}], b(x)\geq 0$. The result remains valid if all the $\leq$ are replaced by $=$. Applying (38) to (24) leads to

$$\iota \leq \hat{\varphi}_r(\mathcal{C}_r) \leq \frac{\iota}{\hat{\alpha}} + \Big(\frac{\iota\hat{\varpi}_r(\hat{\alpha}-1)}{\mathcal{C}_r} - \frac{\iota}{\hat{\alpha}}\Big) e^{\left(\hat{\alpha}\frac{\mathcal{C}_r-\hat{\varpi}_r}{\mathcal{C}_r}\right)}. \tag{39}$$

Thus, the minimum $\hat{\alpha}$ is achieved when all inequalities in (24) and (39) are binding, *i.e.*, $\upsilon = \frac{\iota}{\hat{\alpha}} + \Big(\frac{\iota\hat{\varpi}_r(\hat{\alpha}-1)}{\mathcal{C}_r} - \frac{\iota}{\hat{\alpha}}\Big)e^{\left(\hat{\alpha}\frac{\mathcal{C}_r-\hat{\varpi}_r}{\mathcal{C}_r}\right)}$. This leads to the competitive ratio shown in (37) and the design of $\{\hat{\phi}_r\}_{r\in\mathcal{R}}$.

In the following, we prove the results of (ii). When $\mathcal{R}=1$, $\mathcal{P}_1$ degenerates to the general one-way trading (GOT) problem. The optimal competitive ratio is proved to be $1+\ln(\frac{\upsilon}{\iota})$ [46]–[48]. With $\hat{\alpha}\geq 1, \frac{\upsilon}{\iota}\geq 1$, taking $y\geq 1$ as a substitute for $\hat{\alpha}-1$. Thus

$$\begin{aligned}\hat{\alpha} - 1 - \ln\Big(\frac{\upsilon}{\iota}\Big) &= y - \ln\Big(\frac{\upsilon}{\iota}\Big) \\ &= \ln\Big(y+1-\frac{\iota}{\upsilon}\Big) + \frac{1}{y} - \ln y \quad \triangleright \textit{apply } (37) \\ &:= \mathsf{gap}(y).\end{aligned}$$

Applying $\ln(x)\leq x-1, \forall x\geq 1$ to the logarithm in $\mathsf{gap}(y)$, we have $\mathsf{gap}(y)\leq y+\frac{1}{y}-\ln y-\frac{\iota}{\upsilon}$ given $y\geq\frac{\iota}{\upsilon}$. By analyzing the upper bound of $\mathsf{gap}(y)$, we can easily find that when $y^\star = \frac{\sqrt{5}+1}{2}$, its upper bound is at least $\frac{1}{y^\star} - \ln y^\star$, which directly leads to the result in (ii). □

By the design of $\hat{\phi}_r(\cdot)$, we observe that $\alpha\geq 2$ holds because $\frac{\upsilon}{\iota}\geq 1$. Unsuprisingly, `OnSocMax` has a linear complexity of $\mathcal{O}\big(|\mathcal{N}|\cdot|\mathcal{R}|\big)$ when $\{f_{nr}\}_{n\in\mathcal{N},r\in\mathcal{R}_n}$ are linear and share the same coefficient. In this case, $\hat{\alpha}=2$.

C. Extending to Non-fractional Workloads

`OnSocMax` can be applied to jobs whose workloads are not permitted to be fractionated. For example, each video clip in a video transcoding task must be processed as a whole and cannot be further fragmented. Specifically, in this case, (7) is replaced by

$$x_{nr}\in\{0,\overline{\chi}_{nr}\}, \forall n\in\mathcal{N}, r\in\mathcal{R}_n, \tag{40}$$

where $\overline{\chi}_{nr}=\varrho_n$. To solve the new problem in online settings, we can approximate the marginal cost defined in (17) with $\hat{\phi}_r(\hat{\omega}_r^{(n)}+\hat{\mu}_{nr})\hat{x}_{nr}$, where $\hat{\mu}_{nr}$ is an adjustment term used to estimate the additional cost when task n is assigned to resource unit r. With this substitution, the conditions (c) in step 6 of `OnSocMax` is merged into condition (a) or (b), *i.e.*,

$$x_{nr} = \begin{cases} \varrho_n & \phi_r(\hat{\omega}_r^{(n+1)}) + \mu_n - \frac{\beta_{nr}}{C_r} < f'_{nr}(\overline{\chi}_{nr}) \\ 0 & \phi_r(\hat{\omega}_r^{(n+1)}) + \mu_n - \frac{\beta_{nr}}{C_r} > f'_{nr}(0). \end{cases} \tag{41}$$

This approach still follows the idea of (14) and the KKT conditions. `OnSocMax` achieves the same competitive ratio as shown in **Theorem 2**.

D. Implementation Concerns

In the CMEC system, jobs are processed by invoking cloud functions across different resource units, which relies on the multi-tenant hardware sharing technique such as VM-like isolation. The approach adopted by AWS Lambda is maintaining an active pool of computing instances that have been used to run functions beforehand and are maintained to serve future invocations. In addition, considering that the workloads of one job are dispatched to different resource units across a time window, an efficient communication mechanism is required. A long-running VM-based rendezvous server facilitated by a coordinator can be adopted to relay packets between cloud functions. Based on these techniques, `OnSocMax` can be easily plugged into the API gateway and triggered at the beginning of each time slot. Thus, the process of online workload scheduling by `OnSocMax` is shown in Fig 6.

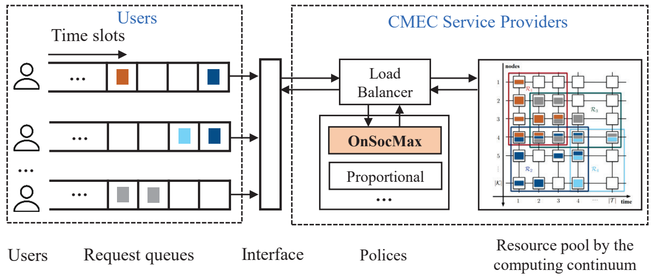


Fig. 6. Online workload scheduling in the computing continuum.

V. Experimental Results and Discussion

Simulations are performed to evaluate the effectiveness and robustness of `OnSocMax`. Firstly, the performance of `OnSocMax` is verified against several handcrafted benchmarking policies on the achieved social welfare. Then, the robustness of `OnSocMax` is analyzed under several different system settings.

A. Experimental Setup

We consider a cluster with 10 computing instances in the time horizon of 24 time slots. The processing capacity of computing instances are generated from an *i.i.d.* Gaussian $\mathcal{N}(\mu = 20, \sigma = 2)$. By setting τ as 60 minutes, the time horizon represents one day. We set the total number of transcoding jobs as 20. The number of job arrivals in each time slot follows a Poisson distribution with a mean of 2.03 jobs, which is independent of other time slots in this day. The deadline of each job is calculated based on the arrival time and the maximum service duration, where the latter is generated from an exponential distribution with a mean of 4 time slots (2 hours). Each job's workload size is generated from a Normal distribution $\mathcal{N}(\mu = 18, \sigma = 3)$. The parallelism bound of each job is generated from a Normal distribution $\mathcal{N}(\mu = 7, \sigma = 1)$.

The utility of job n is set as a zero-startup, non-decreasing concave function. We study f_{nr} in three cases, including linear function, logarithmic function, and polynomial function. Specifically, for each $n \in \mathcal{N}$, $r \in \mathcal{R}_n$,

$$f_{nr}(x) = \begin{cases} ax & \textit{linear} \\ a\log(x+1) & \textit{log} \\ a\sqrt{x} & \textit{poly}, \end{cases}$$

where the coefficient a is generated from a uniform distribution in $[1, 3]$. Similarly, the pricing parameter β_{nr} in $g_{nr}(\cdot)$ is generated from the uniform distribution in $[0.1, 0.5]$.

B. Simulation Results

This part shows the simulation results of `OnSocMax` and its compared algorithms. We first show the validation of `OnSocMax`'s solution as proved in Section IV-B. Then, we show the superiority and robustness of `OnSocMax`. It is compared with three handcrafted online algorithms. The first algorithm is called Max-First, where each computing instance always serves the job with the highest *myopic social welfare*, *i.e.*, the sum of the utility of the job and the revenue for serving it each time slot (subject to the processing capacity of instances and the parallelism bound of jobs). The second algorithm is called Equal-Share, where each computing instance serves each job with equal opportunity within capacity limits. The third one is JSQ, where jobs are assigned to each computing instance with the shortest queue.

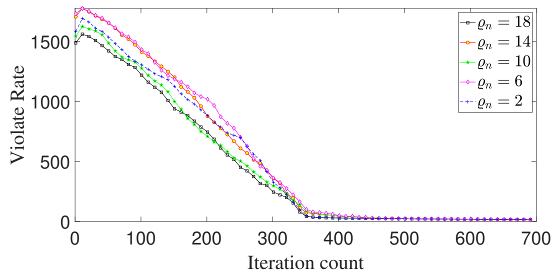


Fig. 7. Violate rate of `OnSocMax` under different iterations.

Fig. 7 shows the violate rate of `OnSocMax` in each iteration. The violate rate denotes whether the online solution $\{\hat{\boldsymbol{x}}_n\}_{n\in\mathcal{N}}$ is satisfied to constraints (4), (5), (7) of $\mathcal{P}_3$, where a lower value indicates a valid solution. It is shown that `OnSocMax` can always find a satisfactory solution after 600 iterations under different workloads. Figs. 8 and 9 show the social welfare of each job achieved by `OnSocMax` and `Max-Fist`, respectively. The darker color indicates the higher social welfare achieved. It is shown that the social welfare of each job obtained by `OnSocMax` is more evenly distributed. On the contrary, `Max-Fist` is more decentralized, *i.e.*, the social welfare of job 4 is 137.37, and that of job 6 is 44.49. This resulted in some tasks receiving much longer service times, causing inequities. The statistics also reflect this phenomenon. The interquartile range reflects data decentralization, where a larger value represents more decentralized data. `OnSocMax` obtains an interquartile range of 14.52, which is much lower

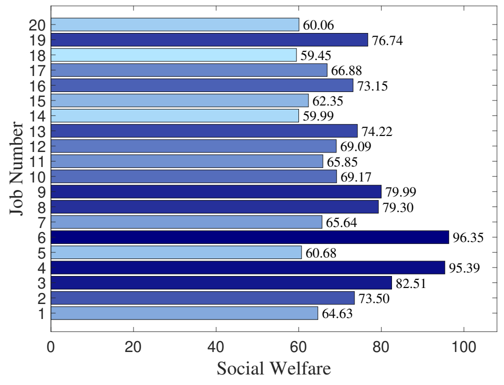


Fig. 8. Welfare of each job achieved by `OnSocMax`.

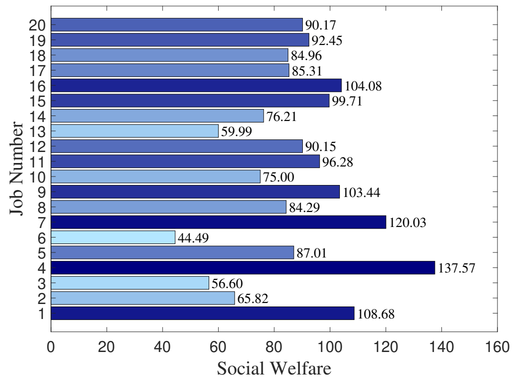


Fig. 9. Welfare of each job achieved by `Max-First`.

than `Max-Fist`'s interquartile range of 25.96, proving the fairness of `OnSocMax`.

Fig. 10 shows the social welfare achieved by four algorithms under different service duration and parallelism bound settings. It is shown that all the algorithms achieve higher social welfare when the service duration and parallelism bound increases. This is because when the computational capacity of computing instances is sufficient, increasing the service duration and the parallelism bound of jobs can increase job opportunities to be fully served. In that case, the utility of each job rises, leading to higher social welfare of the CMEC system. Despite this, `OnSocMax` always performs the best among the compared algorithms. `Max-First` is designed to serve higher myopic social welfare jobs first. Thus, it achieves higher social welfare than `Equal-Share` and `JSQ`. However, it focuses on the optimal choice within each time slot and does not consider tasks arriving in the future and the demand for resources from these tasks. As a result, it may overuse certain computing instances, reducing the overall social welfare.

`JSQ` obtains the lowest social welfare in linear and polynomial utility functions. This is because the service rates of the computing instances are heterogeneous, and jobs have different deadlines and parallelism constraints. In this case, `JSQ` assigns jobs to the shortest queues that may not be suitable for processing, resulting in jobs not completed on time or low resource utilization. In addition, `JSQ` only focuses on the queue length of jobs without considering the utility of each job, which affects the overall social welfare. It is worth noting that `JSQ` achieves higher social welfare than `Max-First` and `Equal-Share` when the service duration is low in Fig. 10(c). The logarithmic utility function is sensitive to tasks with smaller workloads due to its diminishing marginal returns property. Specifically, the logarithmic utility function can provide higher utility for those tasks that can be completed in a shorter time. However, utility growth becomes very slow for tasks that take a long process. For tasks with smaller workloads, the reduction of queuing time is crucial, and by assigning tasks to the shortest queue, JSQ can minimize the queuing time and process tasks with smaller workloads quickly. Thus, JSQ can achieve higher utility gains by completing these tasks rapidly, achieving higher social welfare. By solving a series of pseudo-social welfare maximization problems, `OnSocMax` considers the allocation of resources over longer time horizons and aims to maximize the cumulative social welfare over the entire time slot. It utilizes a marginal cost estimation technique that dynamically adjusts the costs of different resource units to better reflect the actual value and scarcity of resources. This mechanism helps to avoid wastage of resources and ensures that tasks are processed at the right time.

We verify the robustness of `OnSocMax` under different settings of service capacity of computing instances and service demand of jobs. These two variables actually tune the congestion level, *i.e.*, the coverage rate of service demands from different angles. The experimental results are shown in Figs. 11 and 12. `OnSocMax` performs the best for linear utility functions. This is because the logarithmic and polynomial functions have diminishing returns, which could increase the fluctuation ratio $\frac{\upsilon}{\iota}$. This will cause more jobs served with their marginal costs to fall into the second segment of $\hat{\phi}_r(\omega)$, further leading to a decrease in social welfare. It is shown that `OnSocMax` is robust to the changes in the congestion level, and it can always effectively schedule the online workload in different environment settings.

VI. Conclusion

The computing continuum paradigm integrates edge and cloud resources to support computationally intensive and real-time applications. It derives cloud-assisted mobile edge computing systems with great potential for delivering high-bandwidth and low-latency services to numerous end users. In these systems, it is essential to dispatch jobs to the appropriate backend resources of edge or cloud servers. This work investigates an online workload scheduling problem, considering the system's resource allocation to maximize social welfare. The job we considered is continuous arriving, deadline-sensitive, and has maximum parallelism bound. A model of the resource pool is established to consider both the spatial and temporal resources in the computing continuum and each job can only be dispatched to the resource units that are available to it. Based on the marginal cost estimation technique, an online algorithm `OnSocMax` is designed by following the solutions of several convex pseudo-social welfare maximization problems. It is proved to be α-competitive for an α at least 2. Experimental

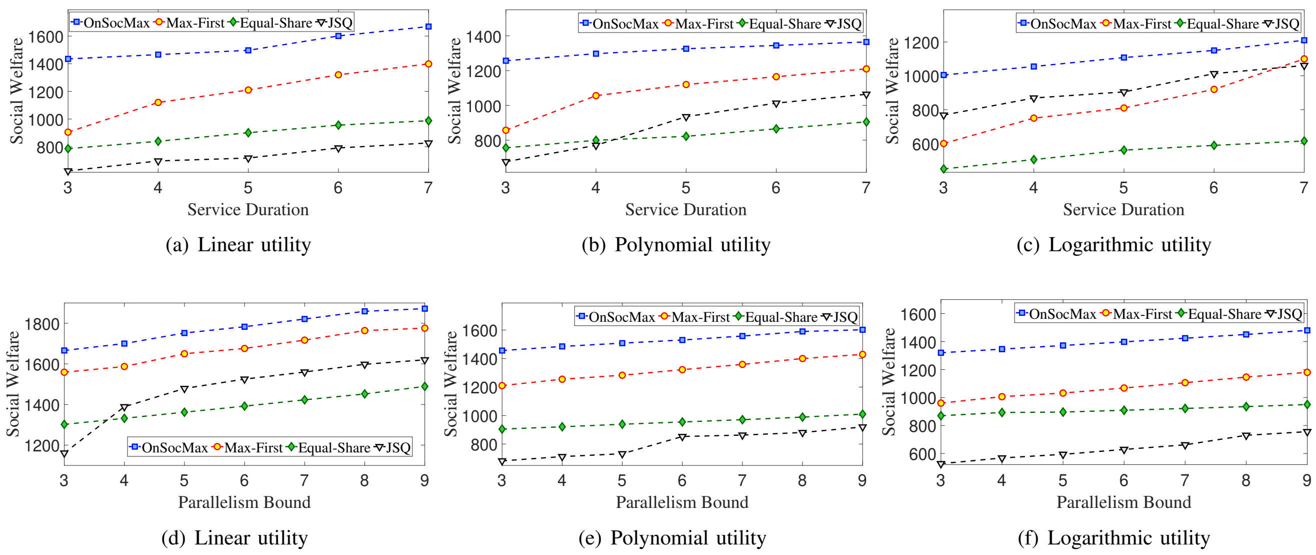


Fig. 10. The social welfare achieved by four algorithms under the change of service duration and parallelism bound.

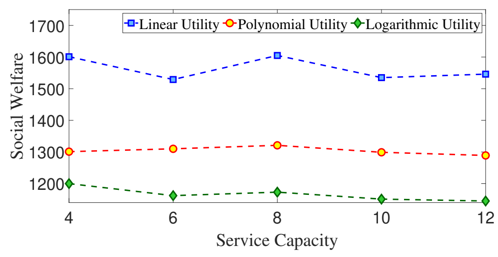


Fig. 11. Achieved social welfare under different service capacities.

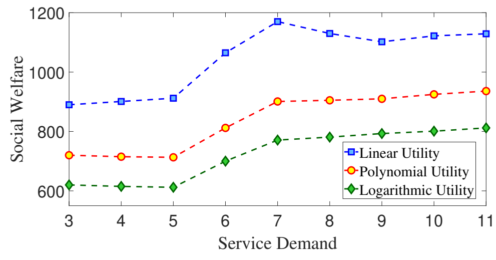


Fig. 12. Achieved social welfare under different service demands.

results show that OnSocMax outperforms three handcrafted online algorithms in maximizing social welfare. In future work, we intend to investigate online workload scheduling for jobs with complex workflows and communication patterns, further enhancing the performance of OnSocMax.

References

[1] I. Čilić, V. Jukanović, I. P. Žarko, P. Frangoudis and S. Dustdar, “QEdgeProxy: QoS-Aware Load Balancing for IoT Services in the Computing Continuum," *2024 IEEE International Conference on Edge Computing and Communications*, Shenzhen, China, 2024, pp. 67–73.

[2] S. Wang, J. Zhang and X. Tan, “PDLC-LIO: A Precise and Direct SLAM System Toward Large-Scale Environments With Loop Closures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 1, pp. 626–637, Jan. 2024.

[3] C. -W. Park, V. Palakonda, S. Yun, I. -M. Kim and J. -M. Kang, “OCR-Diff: A Two-Stage Deep Learning Framework for Optical Character Recognition Using Diffusion Model in Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 11, no. 15, pp. 25997–26000, Aug. 2024.

[4] J. Bi, Z. Wang, H. Yuan, J. Qiao, J. Zhang and M. Zhou, “Self-adaptive Teaching-learning-based Optimizer with Improved RBF and Sparse Autoencoder for Complex Optimization Problems," *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, 2023, pp. 7966–7972.

[5] H. Zhao, S. Deng, Z. Xiang, X. Yan, J. Yin, Schahram. D and A. Zomaya, “Scheduling Multi-Server Jobs With Sublinear Regrets via Online Learning," *IEEE Transactions on Services Computing*, vol. 17, no. 3, pp. 1168–1180, May 2024.

[6] H. Tu, G. Zhao, H. Xu and X. Fang, “Tenant-Grained Request Scheduling in Software-Defined Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4654–4671, Dec. 2022.

[7] Y. Cai, J. Llorca, A. M. Tulino and A. F. Molisch, “Joint Compute-Caching-Communication Control for Online Data-Intensive Service Delivery," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4617–4633, May 2024.

[8] M. Hamdan, E. Hassan, Abdelaziz, A. Abdallah, B. Mohammed, S. Khan, A. Vasilakos and M. Marsono, “A Comprehensive Survey of Load Balancing Techniques in Software-defined Network," *Journal of Network and Computer Applications*, vol. 33, pp. 102856–102870, Jan. 2021.

[9] S. -H. Wu, C. -H. Ko and H. -L. Chao, “On-Demand Coordinated Spectrum and Resource Provisioning Under an Open C-RAN Architecture for Dense Small Cell Networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 673–688, Jan. 2024.

[10] S. Dustdar, V. C. Pujol and P. K. Donta, “On Distributed Computing Continuum Systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4092–4105, Apr. 2023.

[11] J. Bi, Z. Wang, H. Yuan, J. Zhang and M. Zhou, “Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16672–16683, May 2024.

[12] H. Liao, G. Tang, D. Guo, K. Wu and L. Luo, “EV-Assisted Computing for Energy Cost Saving at Edge Data Centers," *IEEE Transactions on Mobile Computing*, vol. 23, no. 9, pp. 9029–9041, Sept. 2024.

[13] Y. Liu, Y. Mao, Z. Liu and Y. Yang, “Deep Learning-Assisted Online Task Offloading for Latency Minimization in Heterogeneous Mobile Edge," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4062–4075, May 2024.

[14] C. -L. Chen, C. G. Brinton and V. Aggarwal, “Latency Minimization for Mobile Edge Computing Networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2233–2247, Apr. 2023.

[15] H. Chen, S. Deng, H. Zhu, H. Zhao, R. Jiang, S. Dustdar and A. Zomaya,

"Mobility-Aware Offloading and Resource Allocation for Distributed Services Collaboration," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2428–2443, Oct. 2022.

[16] A. Mohajer, M. Sam Daliri, A. Mirzaei, A. Ziaeddini, M. Nabipour and M. Bavaghar, "Heterogeneous Computational Resource Allocation for NOMA: Toward Green Mobile Edge-Computing Systems," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1225–1238, Mar. 2023.

[17] Y. Du, J. Li, L. Shi, T. Liu, F. Shu and Z. Han, "Two-Tier Matching Game in Small Cell Networks for Mobile Edge Computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 254–265, Jan. 2022.

[18] X. Liu and L. Ying, "Universal Scaling of Distributed Queues Under Load Balancing in the Super-Halfin-Whitt Regime," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 190–201, Feb. 2022.

[19] S. Bhambay and A. Mukhopadhyay, "Optimal Load Balancing in Heterogeneous Server Systems," *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, Torino, Italy, 2022, pp. 113-120.

[20] W. Weng, X. Zhou and R. Srikant, "Optimal Load Balancing in Bipartite Graphs," *arXiv preprint arXiv: 2008.08830*, 2020.

[21] G. Wu, H. Wang, H. Zhang, Y. Shen, S. Shen and S. Yu, "Mean-Field Game-Based Task-Offloaded Load Balance for Industrial Mobile Edge Computing Systems Using Software-Defined Networking," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 13773–13786, Dec. 2024.

[22] S. Sthapit, J. Thompson, N. M. Robertson and J. R. Hopgood, "Computational Load Balancing on the Edge in Absence of Cloud and Fog," *IEEE Transactions on Mobile Computing*, vol. 18, no. 7, pp. 1499–1512, Jul. 2019.

[23] G. Aumala, E. Boza, L. Ortiz-Avilés, G. Totoy and C. Abad, "Beyond Load Balancing: Package-Aware Scheduling for Serverless Platforms," *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Larnaca, Cyprus, 2019, pp. 282–291.

[24] Z. Xiang, Y. Zheng, D. Wang, J. Taheri, Z. Zheng and M. Guo, "Cost-Effective and Robust Service Provisioning in Multi-Access Edge Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 10, pp. 1765–1779, Oct. 2024.

[25] Z. Zheng and N. B. Shroff, "Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud," *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, CA, USA, 2016, pp. 1–9.

[26] S. Duan, F. Fyu, H. Wu, W. Chen, H. Lu, Z. Dong and X. Shen, "MOTO: Mobility-Aware Online Task Offloading With Adaptive Load Balancing in Small-Cell MEC," *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 645–659, Jan. 2024.

[27] V. Bilò, G. Monaco, L. Moscardelli and C. Vinci, "Nash Social Welfare in Selfish and Online Load Balancing" *ACM Transactions on Economics and Computation*, vol. 10, no. 2, pp. 1–41, Oct. 2022.

[28] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar and A. Y. Zomaya, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[29] H. Yuan, J. Bi, Z. Wang, J. Yang, and Jia Zhang, "Partial and Cost-minimized Computation Offloading in Hybrid Edge and Cloud Systems," *Expert Systems with Applications*, vol. 250, pp. 1–13, Sept. 2024.

[30] H. Yao, R. Ni, H. Amirpour, C. Timmerer and Y. Zhao, "Detection and Localization of Video Transcoding From AVC to HEVC Based on Deep Representations of Decoded Frames and PU Maps," *IEEE Transactions on Multimedia*, vol. 25, pp. 5014–5029, Jun. 2023.

[31] D. P. Bertsekas, "Nonlinear Programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, Jun. 1997.

[32] X. Wu and J. Lu, "Fenchel Dual Gradient Methods for Distributed Convex Optimization Over Time-Varying Networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 11, pp. 4629–4636, Nov. 2019.

[33] T. Yu, R. Zhong, V. Janjic, P. Petoumenos, J. Zhai, H. Leather and J. Thomson, "Collaborative Heterogeneity-Aware OS Scheduler for Asymmetric Multicore Processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1224–1237, May 2021.

[34] J. H. M. Korndörfer, A. Eleliemy, A. Mohammed and F. M. Ciorba, "LB4OMP: A Dynamic Load Balancing Library for Multithreaded Applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 830–841, Apr. 2022.

[35] X. Fu and E. Modiano, "Learning-NUM: Network Utility Maximization With Unknown Utility Functions and Queueing Delay," *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp. 2788–2803, Dec. 2022.

[36] C. Zhang, H. Tan, H. HUang, Z. Han, S. Jiang, G. Li and X. Li, "Online Approximation Scheme for Scheduling Heterogeneous Utility Jobs in Edge Computing," *IEEE/ACM Transactions on Networking*, vol. 31, no. 1, pp. 352–365, Feb. 2023.

[37] X. Yao, X. Yang, Q. Li, C. Qi, X. Kong and X. Li, "UMIM: Utility-Maximization Incentive Mechanism for Mobile Crowd Sensing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 6334–6346, May 2024.

[38] Google Cloud, "Pricing details for the Transcoder API," https://cloud.google.com/transcoder/pricing, 2024.

[39] Amazon Web Services, Inc, "Amazon Elastic Transcoder Pricing," https://aws.amazon.com/elastictranscoder/pricing/, 2024.

[40] Alibaba Cloud, "Alibaba Cloud Media Processing," https://www.alibabacloud.com/product/mts/pricing, 2024.

[41] Tencent Cloud, "Media Processing Service," https://cloud.tencent.com/product/mps/pricing, 2024.

[42] P. Cong, G. Xu, J. Zhou, M. Chen, T. Wei and M. Qiu, "Personality- and Value-Aware Scheduling of User Requests in Cloud for Profit Maximization," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 1991–2004, Jul. 2022.

[43] B. Sun, and A. Zeynali,T. Li, M. Hajiesmaili, A. Wierman and D. Tsang, "Competitive Algorithms for the Online Multiple Knapsack Problem with Application to Electric Vehicle Charging," *Proc. ACM Meas. Anal. Comput. Syst.*, Dec. 2020.

[44] Q. Wang, S. Guo, J. Liu, C. Pan and L. Yang, "Profit Maximization Incentive Mechanism for Resource Providers in Mobile Edge Computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 138–149, Jan. 2022.

[45] J. Bi, Z. Wang, H. Yuan, J. Zhang and M. Zhou, "Self-adaptive Teaching-learning-based Optimizer with Improved RBF and Sparse Autoencoder for High-dimensional Problems," *Information Sciences*, vol. 630, pp. 463–481, Jun. 2023.

[46] X. Tan, B. Sun, A. Leon-Garcia, and Y. Wu and T. Danny, "Mechanism Design for Online Resource Allocation: A Unified Approach," *ACM on Measurement and Analysis of Computing Systems*, pp. 1–46, Jun. 2020.

[47] R. El-Yaniv, A. Fiat, R. Karp and G. Turpin, "Optimal Search and One-way Trading Online Algorithms," *Algorithmica*, vol. 30, no. 1, pp. 101–139, May 2001.

[48] X. Tan, A. Leon-Garcia, Y. Wu and D. H. K. Tsang, "Online Combinatorial Auctions for Resource Allocation With Supply Costs and Capacity Limits," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 4, pp. 655–668, Apr. 2020.