Online Generalized Magician's Problem with Multiple Workers

Ruoyu Wu[1] Wei Bao[1] Ben Liang[2] Liming Ge[1]

[1]School of Computer Science, The University of Sydney, Sydney, NSW, Australia
[2]Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada

Abstract

We study the online Generalized Magician's Problem with Multiple Workers (GMPMW), where tasks arrive sequentially and must be assigned to one of several workers for processing, with each worker consuming a stochastic amount of resources and generating an unknown reward. The system must decide on the acceptance of each task and its assignment to a worker, in order to maximize the accumulated reward within the budget. To address this problem, we propose the Online Worker Assignment (OWA) Algorithm. It optimally solves an optimization problem to balance resource allocation across workers and maintains virtual resource utilization according to the joint evolution of different workers. The competitive ratio of OWA is lower bounded by the closed-form expression $\max\{1/L, c\} \cdot (1 - K^{-\frac{1}{2}})$, where L is the number of workers, K is the resource budget, and c is a constant derived from the problem instance. We perform trace-driven experiments with real-time video analytics, demonstrating the excellent capability of OWA to accommodate multiple workers in GMPMW.

1 INTRODUCTION

The Generalized Magician's Problem (GMP) [Alaei et al., 2013, Alaei, 2014] is a classical online optimization problem, where a decision maker (magician) chooses from a sequence of tasks to process in order to obtain as much reward as possible. Each task has a resource consumption (cost) and a reward initially unknown to the decision maker, which are revealed in stages when the task arrives and is completed.

However, inherent in the original GMP is that there is only a *single worker* processing all accepted tasks. Thus, it fails to consider the complication of many practical scenarios, where the tasks may be processed by different workers, each with different capabilities and resource profiles that influence reward and resource consumption. A generalization of GMP toward multiple workers is well motivated by real-world scenarios in various areas, among which a representative example is task processing in cloud computing. In this application, a cloud provider offers a machine learning (ML) inference service. It receives ML task requests, where different ML models produce different accuracies and require different amounts of energy. The cloud provider needs to decide which task to accept or reject, and which model to use to process each accepted task, in order to maximize the accumulated accuracy. In Section 6 (and Appendix H), this scenario is used as a case study to evaluate our proposed algorithm. Other applications of the GMP with multiple workers include labor outsourcing with different workers and online ad placement with different types of ads. Further details on these applications can be found in Appendix A.

In this paper, we introduce the Generalized Magician's Problem with Multiple Workers (GMPMW): Each task can be processed by one of several workers. Different workers generate different amounts of reward while consuming different amounts of resources. The decision maker must decide on the acceptance of each task and its assignment to a worker, in order to maximize the accumulated reward within a resource budget. To the best of our knowledge, this is the first work to consider multiple workers in GMP.

The GMPMW is substantially more complex than the original GMP, which arises from the need to balance resource consumption across different workers while relying only on limited and incrementally revealed knowledge of tasks – a challenge absent in the single-worker GMP. Likewise, this challenge is absent in other online problems, as it is unique to GMPMW that we consider multiple stages of observability of a task's cost, progressing from zero initial knowledge to a probability distribution upon task arrival, and finally the exact value upon task completion (see details in Section 2).

To overcome these challenges, we develop the Online Worker Assignment (OWA) algorithm to employ a balanced probability-fitting approach. We first balance the workers' resource consumption by optimally solving a problem with a non-convex constraint, which then guides online resource allocation through a set of assignment guarantees. Then, OWA tracks the virtual resource consumption, which captures the joint evolution of resource usage across workers and serves as the basis for planning the overall resource usage across tasks. Consequently, OWA balances resource consumption across workers and maximizes resource utilization to effectively handle uncertain rewards and fluctuating resource demands.

Main Results:

- OWA establishes a novel framework for addressing the GMPMW. We derive the competitive ratio α of OWA, along with a closed-form lower bound $\alpha' = \max\{1/L, c\} \cdot (1 - K^{-\frac{1}{2}})$, where L is the number of workers, K is the resource budget, and c is a constant derived from the problem instance. When there is only one worker, i.e., $L = 1$, this lower bound is consistent with the previous best result on the GMP Alaei et al. [2013]. Furthermore, we show that α can be reached in certain problem instances, so it is a tight performance bound.
- We prove that when the reward lower bound for each worker is 0, OWA is asymptotically optimal, meaning that α approaches the best competitive ratio as the resource budget increases. This result underscores the effectiveness of OWA's design and suggests its superiority compared with alternative algorithms.
- For the numerical case study, we perform trace-driven experiments on real-time video analytics over edge devices. These experiments validate the theoretical result and demonstrate OWA's efficiency in leveraging multiple workers in GMPMW.

2 RELATED WORK

A distinct feature of GMP and GMPMW is their staged progression of task cost observability—initially unknown, partially observable as a distribution upon task arrival, and fully revealed as an exact value upon task completion. Figure 1 compares the progression of task cost observability for different problems. Task cost observability may increase at four key instants: before task arrival, upon arrival, upon discarding, and after processing, categorized into three levels: Unknown (no information), Distribution (distribution known), and Value (value known).

We compare GMP and GMPMW (red) against the Online Generalized Assignment Problem (OGAP), Online Stochastic Generalized Assignment Problem (OSGAP), Online Knapsack Problem (OKP), Online Stochastic Knapsack Problem (OSKP), and the Bandit with Knapsack (BwK) problem, including its Full Feedback (BwK-F) and Bandit Feedback (BwK-B) variants. Unlike other problems that experience only two levels of task cost observability, GMP and GMPMW are the only ones that undergo all three levels. We elaborate below on these differences, with a more detailed comparison provided in Appendix B.

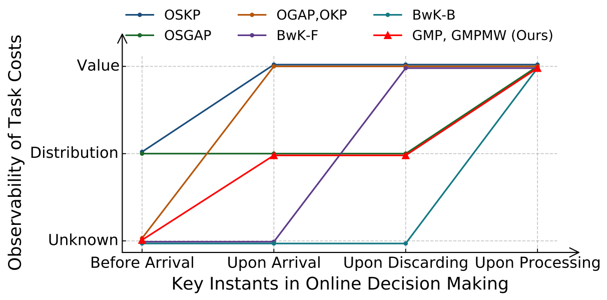


Figure 1: Progression of task cost observability in online problems.

2.1 GMP, OGAP, AND OSGAP

GMP was originally studied by Alaei et al. [2013] with a competitive ratio of $1 - K^{-1/2}$ achieved, where K is the resource budget. The Magician's Problem (MP) Alaei [2014] is a special case of GMP with random 0-1 cost. Srinivasan and Xu [2022] studied a variant of GMP considering unknown distributions of resource consumption. GMP has then been applied in e-commerce [Amil et al., 2025] and transportation [Jiang and Samaranayake, 2022]. However, none of the above works can accommodate the multiple workers in GMPMW.

GMP has been adopted to tackle other online problems, such as OSGAP [Alaei et al., 2013]. In OSGAP, each task belongs to a type drawn from a known distribution. Upon arrival, the task's type is revealed, specifying the reward and the cost distribution. Yoshinaga and Kawase [2023] extended OSGAP to consider a more limited resource budget. Liu et al. [2023] and Li et al. [2023] studied OGAP with adversarial task rewards and costs. However, as shown in Figure 1, both OSGAP and OGAP follow a similar observability progression: the exact task cost is revealed upon arrival (OGAP) or upon processing (OSGAP) without an intermediate level. Consequently, none of these problems considers the full progression of task cost observability involving all three levels as in GMP and GMPMW.

2.2 OKP, OSKP, AND BWK

OKP [Zhou et al., 2008, Böckenhauer et al., 2014] assumes that the decision-maker initially has no knowledge of either task rewards or costs, and OSKP assumes that the decision-maker knows the distribution [Papastavrou et al., 1996, Dean et al., 2008] or the exact value [Jiang et al., 2022] of task costs. However, the exact task cost in both problems is revealed upon arrival. In BwK [Badanidiyuru et al., 2018, Immorlica et al., 2022, Drago et al., 2024], the task's reward and cost are not known to the decision-maker and are revealed after processing the task (BwK-B) or after the task is either processed or discarded (BwK-F). To summarize, as illustrated in Figure 1, none of OKP, OSKP, or BwK consider the full progression of task cost observability as in GMP and GMPMW.

2.3 OTHER ONLINE OPTIMIZATION PROBLEMS

Recently, Feldman et al. [2021] introduced the Online Contention Resolution Scheme (OCRS) as a rounding scheme for solving the Online Bayesian Optimization Problem (BOP) [Chawla et al., 2010, Gupta and Nagarajan, 2013, Feldman et al., 2021, Jiang et al., 2022]. However, all of these works consider deterministic task costs known to the decision-maker at the beginning. Moreover, the lack of prior knowledge about task rewards and costs prevents the construction of a linear relaxation for solving GMP via OCRS and necessitates fundamentally different approaches.

There are additional online optimization problems that have structural differences from GMP and GMPMW, such as the One-Way Trading Problem (OTP) [El-Yaniv et al., 2001, Lin et al., 2019, Cao et al., 2020], the Online Bipartite Matching Problem [Mehta et al., 2013, Dickerson et al., 2021, Wu et al., 2023], and the online Pandora's Box problem [Esfandiari et al., 2019, Boodaghians et al., 2020, Gatmiry et al., 2024]. We discuss these differences in more detail in Appendix B.4.

3 THE MAGICIAN'S PROBLEM WITH MULTIPLE WORKERS

In this section, we first present a formal description of the GMPMW, and then discuss the online environment and our objective of the algorithm design, including a formal definition of the competitive ratio adopted in GMP.

3.1 PROBLEM FORMULATION

A sequence of T tasks arrives at the decision maker one at a time. For each task, the decision maker may choose one of L different workers to process it, indexed by l. When task t arrives, the decision maker makes irrevocable decisions $x_{t,l} \in \{0,1\}$ for each worker l, determining whether to assign the task to worker l (i.e., $x_{t,l} = 1$) or to reject it (i.e., $x_{t,l} = 0$ for all l).

If task t is processed by worker $l \in [L]$, it will generate a reward $u_{t,l}$. Otherwise, task t is discarded and cannot be processed in the future. The reward $u_{t,l}$ of processing task t with worker l is unobservable, determined by the hidden factors (e.g., user satisfaction), and it is revealed only after the task is completed. Although $u_{t,l}$ is not known in advance, it varies within a range $[\underline{u}_l, \overline{u}_l]$, and the upper bound $\overline{u}_l$ and the lower bound $\underline{u}_l$ are known to the decision maker. We assume that $u_{t,l} \geq 0$.

The decision maker has a total resource budget of K for the workers. When task t arrives, the probability distribution $\mathcal{R}_{t,l}$ for the resource consumption of each worker l to process this task becomes known. This progressively improving observability of task cost reflects practical scenarios where resource consumption is influenced by measurable external factors (e.g., temperature) or observable task attributes (e.g., size), allowing for estimation upon the task's arrival. When the decision maker assigns task t to worker l, the worker consumes an amount of resource $r_{t,l}$ that is drawn from $\mathcal{R}_{t,l}$, which is then deducted from the remaining amount of the total resource budget. Without loss of generality, we assume that $r_{t,l} \in (0,1]$. We further assume that $\mathcal{R}_{t,l}$ has a probability density function (PDF) $g_{t,l}(\cdot)$ that is continuous, along with the cumulative distribution $G_{t,l}(\cdot)$.

We consider the hard budget constraint: If processing a task exceeds the remaining budget, the task will be discarded, and the consumed resource will not be recovered. Following the conventional large-scale setting proposed by [Alaei et al., 2013, Alaei, 2014] for GMP, we assume that $\frac{1}{L}\sum_{l\in[L]}\sum_{t\in[T]}\mathbb{E}[r_{t,l}] \leq K$, i.e., the decision maker's resource is, on average, sufficient to process all tasks. The large-scale setting assumption of the GMPMW accounts for real-world engineering and operational problems where a reasonable budget is allocated to the decision-maker.[1] We also assume that $K \geq 1$, which is also conventionally adopted in GMP.

Our objective is to maximize the accumulated reward over the incoming tasks under the resource budget constraint. We formulate our optimization problem as follows:

$$\max_{x_{t,l},\forall t,l} \quad \sum_{t\in[T]}\sum_{l\in[L]} x_{t,l}u_{t,l}, \tag{1}$$

$$\text{s.t.} \quad \sum_{t\in[T]}\sum_{l\in[L]} x_{t,l}r_{t,l} \leq K, \tag{2}$$

$$\sum_{l\in[L]} x_{t,l} \leq 1, \forall t \in [T], \tag{3}$$

$$x_{t,l} \in \{0,1\}, \forall l \in [L], t \in [T], \tag{4}$$

$$r_{t,l} \sim \mathcal{R}_{t,l}, \forall l \in [L], t \in [T], \tag{5}$$

where constraint (2) indicates the limited resource budget,

[1]For example, in project portfolio selection or maintenance scheduling, organizations often plan a budget that is, in expectation, sufficient to cover most (if not all) tasks, while the resources must be allocated wisely.

constraints (3) specify that at most one worker is assigned to process each task, constraints (4) give the decision space, and (5) specify the distributions of resource consumption.

3.2 ONLINE ENVIRONMENT AND COMPETITIVE RATIO

In GMPMW, each task sequence I is arranged by an adversary, with T, $\{u_{t,l}\}$, and $\{\mathcal{R}_{t,l}\}$ chosen by the adversary. A task sequence I is random due to the distributions $\{\mathcal{R}_{t,l}\}$. We define Ω^I as the sample space of I and $i \in \Omega^I$ as a sample path of the task sequence I. A sample path i has a set of $r_{t,l}$ generated from $\{\mathcal{R}_{t,l}\}$ in its task sequence.

The decision maker does not know T, $\{u_{t,l}\}$, or $\{\mathcal{R}_{t,l}\}$ in advance since these values are chosen by the adversary. The length T of the task sequence is revealed when no more tasks arrive to the decision maker; the probability distribution of resource consumption $\mathcal{R}_{t,l}$ is revealed with its PDF $g_{t,l}(\cdot)$ when task t arrives, which as explained previously is a distinct feature of GMP and GMPMW; and the reward $u_{t,l}$ and the exact resource consumption $r_{t,l}$ are known to the decision maker only after task t is processed by worker l. Note that $u_{t,l}$ and $r_{t,l}$ remain unknown to the decision maker if task t is not processed by worker l.

For a given sample path i, we denote the performance of an online algorithm ALG as ALG(i) and denote the optimal performance of the offline algorithm as OPT(i). We assume that i is fully known to OPT in advance. We denote the average performance of the online algorithm ALG over all sample paths i of the task sequence I as $\mathbb{E}_{i\sim I}[\text{ALG}(i)]$. To assess the performance of an online algorithm in this paper, we use the competitive ratio as the primary metric. Here, ALG is *α-competitive* if

$$\mathbb{E}_{i\sim I}[\text{ALG(i)}] \geq \alpha \max_{i\in\Omega^I}[\text{OPT}(i)], \forall I \in \mathcal{I}, \tag{6}$$

where $\mathcal{I}$ is the set of all possible task sequences. This definition of the competitive ratio is previously adopted by Alaei et al. [2013], Alaei [2014], Srinivasan and Xu [2022]. Note that this definition is stronger than another commonly used definition [Buchbinder et al., 2009]:

$$\mathbb{E}_{i\sim I}[\text{ALG}(i)] \geq \alpha \mathbb{E}_{i\sim I}[\text{OPT}(i)], \forall I \in \mathcal{I}, \tag{7}$$

as (6) implies (7).

4 ONLINE WORKER ASSIGNMENT (OWA) ALGORITHM DESIGN

In this section, we present our Online Worker Assignment (OWA) framework for solving GMPMW. OWA is an online algorithm that addresses the complicated trade-off between accumulating rewards and balancing limited resources among multiple workers. It requires fully leveraging the progressively improved task cost observability, making its design significantly more challenging and distinct from existing solutions for the original GMP and other online problems.

The OWA algorithm (summarized in Alg. 1) is a balanced probability-fitting approach consisting of four main phases: Pre-Calculation, Worker-Assignment, Processing, and Baseline-Calibration. ① Before the first task arrives, OWA enters the pre-calculation phase (Line 3), where it calculates an *assignment guarantee* γ_l^* for each worker l based on reward bounds and the resource budget (Eqs. (8)–(11)), which later guides the decision-making in the online process by balancing online resource consumption across workers (Line 7). ② When task t arrives, we enter the worker-assignment phase (Lines 6–8), where OWA compares the *resource utilization level* Θ_t against a *resource utilization baseline* θ_t to decide whether to accept this task. ③ In the processing phase (Lines 10–17), the task is processed or discarded according to the decision. ④ In the baseline-calibration phase (Lines 19–23), OWA first updates the *virtual resource consumption* to capture the long-term resource availability and the joint evolution of different workers, accounting for both the randomness of task cost and the randomization of online decisions. It then derives a new baseline θ_{t+1} for the next decision, carefully fitting the probability of processing the next task to the long-term resource availability, striking a balance between accumulating rewards and preserving sufficient resources for future high-reward tasks.

4.1 PRE-CALCULATION PHASE

Before the first task arrives, OWA starts in the pre-calculation phase to determine an assignment guarantee γ_l^* for each worker l. It is the optimal solution to the following problem:

$$\mathtt{P}_1 \quad \max_{\gamma_l,\forall l} \quad \min_l \frac{\sum_{l'\in[L]\setminus\{l\}} \gamma_{l'}\underline{u}_{l'}}{\overline{u}_l} + \gamma_l \tag{8}$$

$$\textbf{s.t.} \quad K \geq \frac{1}{(1-\sum_{l\in[L]}\gamma_l)(1-\max_l \gamma_l L)}, \tag{9}$$

$$\max_l \gamma_l < \frac{1}{L}, \tag{10}$$

$$\gamma_l \geq 0,\ \forall l. \tag{11}$$

This optimization problem captures the limited prior knowledge of task rewards and costs while accounting for the impact of each decision on future resource dynamics, ensuring that the assignment guarantees optimally balance resource consumption across workers in online decision-making. The reason why OWA sets the assignment guarantee γ_l^* in this way will be explained in detail later in Section 5, where we derive the competitive ratio of OWA.

Since the decision maker knows L, K, $\underline{u}_l$, and $\overline{u}_l$, the optimization problem $\mathtt{P}_1$ is solved in an offline manner be-

Algorithm 1 Online Worker Assignment (OWA) Algorithm

```
INPUT: $L$, $K$, $\{\underline{u}_l\}$ and $\{\bar{u}_l\}$
1: $t \leftarrow 1, \theta_1 \leftarrow 0, \Theta_1 \leftarrow 0, x_{t,l} \leftarrow 0, \phi_1 \leftarrow 1$
2: # Pre-Calculation Phase
3: Calculate $\gamma_l^*$ by solving optimization problem $\mathtt{P}_1$ in Eqs. (8)–(11).
4: while task $t$ arrives ($g_{t,l}$ is observed) do
5:    # Worker-Assignment Phase
6:    if $\Theta_t \leq \theta_t$ and $K - \Theta_t \geq 1$ then
7:       Select a worker $l$ with probability $\gamma_l^*/\phi_t$ and set $x_{t,l} \leftarrow 1$.
8:    end if
9:    # Processing Phase
10:   if $\sum_l x_{t,l} = 1$ then
11:      The selected worker $l$ processes task $t$.
12:      (Reward $u_{t,l}$ and resource consumption $r_{t,l}$ are revealed).
13:      $\Theta_{t+1} \leftarrow \Theta_t + r_{t,l}$
14:   else
15:      Discard task $t$
16:      $\Theta_{t+1} \leftarrow \Theta_t$
17:   end if
18:   # Baseline-Calibration Phase
19:   if $t = 1$ then
20:      Initialize $h_1(\cdot)$ as $\delta(\cdot)$.
21:   end if
22:   Calculate $h_{t+1}(\cdot)$ by Eq. (12).
23:   Calculate $\theta_{t+1}$ and $\phi_{t+1}$ by Eqs. (15)–(16).
24:   $t \leftarrow t + 1$
25: end while
```

fore the tasks arrive. Note that although problem $\mathtt{P}_1$ has a non-convex constraint (9), its optimal solution can still be obtained using standard convex optimization solvers, as detailed in Section 5. Throughout this paper, we denote the optimal value of the objective of $\mathtt{P}_1$ by $\mathtt{P}_1(\{\gamma_l^*\})$, such that $\mathtt{P}_1(\{\gamma_l^*\}) = \min_l (\sum_{l' \in [L] \setminus \{l\}} \gamma_{l'}^* \underline{u}_{l'})/\bar{u}_l + \gamma_l^*$.

4.2 WORKER-ASSIGNMENT PHASE

When task t arrives, OWA starts the worker-assignment phase (Lines 5–7). For all tasks, we use $\Theta_t = \sum_{\tau<t} \sum_l x_\tau r_{\tau,l}$ to record the current resource utilization level at the arrival of task t. If the remaining amount of resource $K - \Theta_t$ is less than 1, OWA discards task t to avoid exceeding the resource budget and causing an infeasible solution, due to the random nature of the resource consumption. If the remaining amount of resource $K - \Theta_t$ is at least 1 and $\Theta_t \leq \theta_t$, OWA selects a worker l with probability γ_l^*/ϕ_t and sets $x_{t,l} = 1$, or discards task t with probability $1 - \sum_l \gamma_l^*/\phi_t$ and sets $x_{t,l} = 0$ for all l. Here, ϕ_t is an offset parameter initialized to 1 and updated in the baseline-calibration phase, and its update rule will be discussed shortly in Section 4.4.

4.3 PROCESSING PHASE

After the decision $\{x_{t,l}\}$ is made, OWA enters the processing phase. In this phase, task t is processed by the selected worker l or discarded, and Θ_t is updated accordingly. If task t is processed by worker l, its reward $u_{t,l}$ is revealed and received by the decision maker, and its consumption $r_{t,l}$ is realized from the probability distribution $\mathcal{R}_{t,l}$ and is counted toward the current resource utilization level in Θ_t (Line 13). Otherwise, the reward $u_{t,l}$ and the resource consumption $r_{t,l}$ of the task t for each worker l remain unknown. The decision maker receives no reward from this task, and Θ_t does not change toward Θ_{t+1} (Line 16).

4.4 BASELINE-CALIBRATION PHASE

In the Baseline-Calibration Phase, given the observed PDF $g_{t,l}(\cdot)$ of the resource consumption of each worker, OWA generates the new resource utilization baseline θ_{t+1} and the new offset ϕ_{t+1} (Lines 18–23). Note that this calculation does not depend on whether the task t is processed. This calculation is based on the *virtual resource consumption* $h_t(w)$, which evaluates the impact of all past decisions on future resource consumption by considering all potential task acceptances, task assignments to different workers, and the corresponding resource consumption. It is initialized as the Dirac delta function $h_1(w) = \delta(w)$, representing the initial budget before any consumption, and is updated as

$$\begin{aligned} h_{t+1}(w) =& (1 - \sum_{l \in [L]} \gamma_l^*/\phi_t) h_t(w) + (\sum_{l \in [L]} \gamma_l^*/\phi_t) \\ & \times \left[\overline{h}_t(w) * \overline{g}_t(w) + h_t(w) - \overline{h}_t(w)\right]. \end{aligned} \tag{12}$$

In Eq. (12), the first summation term captures the impact of potentially discarding task t on future resource consumption, while the second term corresponds to the decision to accept it. In the second summation term in Eq. (12), $\overline{h}_t$ is the truncated resource utilization function

$$\overline{h}_t(w) = \begin{cases} h_t(w), \ w \leq \theta_t, \\ 0, \ w > \theta_t, \end{cases} \tag{13}$$

capturing the impact of baseline θ_t on the decision making, $\overline{g}_t$ is the average resource consumption PDF calculated by

$$\overline{g}_t(w) = (1/\sum_{l \in [L]} \gamma_l^*) \sum_{l \in [L]} \gamma_l^* g_{t,l}(w), \tag{14}$$

capturing the balanced resource consumption across workers while incorporating the newly improved observability (i.e., distribution) of task costs, and $*$ is the convolution operator, capturing the potential future dynamics of resource consumption caused by processing task t. In this way, the virtual resource consumption h_{t+1} captures the stochastic nature of actual resource consumption and the long-term resource availability, accounting for both the randomness of task costs and the randomization in decision-making. We then calculate θ_{t+1} and ϕ_{t+1} (Line 23) as follows:

$$\theta_{t+1} = \arg\min_{w} \left\{ \int_{-\infty}^{w} h_{t+1}(v)dv \geq \sum_{l\in[L]} \gamma_l^* \right\}, \quad (15)$$

$$\phi_{t+1} = \int_{-\infty}^{\theta_{t+1}} h_{t+1}(v)dv, \quad (16)$$

in order to ensure that the baseline θ_{t+1} and offset ϕ_{t+1} carefully "fit" the resource consumption of processing task $t+1$ to the long-term resource availability, balancing immediate resource utilization with preserving resources for the future.

4.5 OWA ALGORITHM COMPLEXITY ANALYSIS

We provide a detailed complexity analysis of OWA in Appendix D and offer a summary here. The complexity of the pre-calculation phase is $\mathcal{O}(L^3)$ when the optimal solution to $\mathtt{P}_1$ is obtained by the *barrier method* [Boyd and Vandenberghe, 2004]. The complexity of the worker-assignment phase is $\mathcal{O}(1)$. In the baseline-calibration phase, the random variables $\mathcal{R}_{t,l}$ can be continuous, with continuous PDFs $\{g_{t,l}\}$. In practice, we can discretize them with arbitrary granularity. Let D be the number of levels used for discretization (a larger D implying higher accuracy). The complexity is $\mathcal{O}(D\log(D))$ in this phase. We note that the pre-calculation phase is performed offline before online tasks arrive. Only the processing phase and the baseline-calibration phase are performed in an online manner. Accordingly, the overall complexity of OWA for each task is $\mathcal{O}(D\log(D))$.

5 THEORETICAL ANALYSIS

In this section, we analyze the competitive ratio of the OWA algorithm and show that it is tight and approaches the maximum possible competitive ratio as the budget K increases. Before that, we first show that the optimization problem $\mathtt{P}_1$ can be solved by standard convex optimization solvers.

Theorem 1. *(Convex Equivalence) There exists a convex optimization problem* $\mathtt{P}'$*, such that its optimal solution is also the optimal solution to* $\mathtt{P}_1$.

Proof. (Sketch of Proof) We first transform the objective function into an equivalent form. We then show that there exists an optimal solution that satisfies the quality of a key constraint, so we can further transform the problem into a convex optimization problem. Consequently, the optimal solution of this convex problem is also optimal for $\mathtt{P}_1$. The complete proof is provided in Appendix C. □

5.1 COMPETITIVE RATIO OF THE OWA ALGORITHM

To analyze the competitive ratio of the OWA algorithm, we will lower-bound the probability of processing each task t by each worker l by the assignment guarantee γ_l^*. To show that OWA avoids the situation where the tasks can no longer be accepted because the remaining amount of resource falls below 1, leading to a 0 probability of processing subsequent tasks, we proactively identify a "safe range" for $\{\gamma_l^*\}$. This ensures that the resource remains sufficiently distant from the stop line. We start with a definition of resource sufficiency in Definition 1.

Definition 1. *We say that the resource sufficiency condition is satisfied if the following statement is true: For each task t,* $K - \Theta_t \geq 1$ *if* $\Theta_t \leq \theta_t$.

The meaning of Definition 1 is that, if the current resource consumption $\Theta_t \leq \theta_t$, we guarantee that the remaining amount of the resource is greater than or equal to 1, allowing processing of the future tasks. We next show that the computed γ_l^* from solving $\mathtt{P}_1$ in Eqs. (8)–(11) satisfies the resource sufficiency condition in Definition 1.

Theorem 2. *(Sufficient Resource) With* γ_l^* *from solving* $\mathtt{P}_1$ *in Eqs. (8)–(11), the OWA algorithm satisfies the resource sufficiency condition.*

Proof. (Sketch of Proof) To prove Theorem 2, we require several lemmas. First, we define H_t as the anti-derivative of h_t and show $H_{t+1}(\theta_t) \leq H_t(\theta_t)$ for every $t \in [T]$. Second, we show that $\theta_t \leq \theta_{t+1}$ for every t. Third, we define $\overline{\Theta}_t = \min_w\{\int_{-\infty}^{w} h_t(v)dv = 1\}$ and show that $\overline{\Theta}_{t+1} \leq \theta_t + 1$ for every t. Fourth, we show that $\theta_t \leq \theta_{t+1} < \theta_t + 1$ for every t. Finally, we show that constraint (9) implies that $K \geq \theta_t + 1$, which in turn leads to the resource sufficiency condition. The complete proof is given in Appendix E.1. □

As a result of Theorem 2, OWA will accept a task and process it by some worker l as long as the current resource consumption Θ_t is less than or equal to the target θ_t. Using such an approach, we can bound from below the probability that each task t will be processed by each worker l.

Theorem 3. *(Worker Assignment Probability Lower Bound) For all l, the probability of processing each task t by worker l is no less than* γ_l^*.

Proof. (Sketch of Proof) To prove Theorem 3, we first show a relation between $h_t(w)$ and the distribution of Θ_t over all sample paths. Using this relation, the design of the OWA algorithm, the update rule of $h_t(w)$, and Theorem 2, we arrive at the theorem's conclusion. The complete proof is given in Appendix E.2. □

Theorem 3 leads to the competitive ratio of OWA:

Theorem 4. *(Competitive Ratio) The competitive ratio achieved by the OWA algorithm is at least* $\alpha = \mathtt{P}_1(\{\gamma_l^*\})$.

Proof. (Sketch of Proof) We use the results of Theorem 3 to analyze the competitive ratio of the OWA algorithm. We find that when the probability of processing each task t by worker l is lower bounded by γ_l^*, the competitive ratio of the OWA algorithm is at least $\mathtt{P}_1(\{\gamma_l^*\})$. The complete proof is given in Appendix E.3. □

Since in general we only have a numerical solution to the optimization problem $\mathtt{P}_1$ and its optimal value $\mathtt{P}_1(\{\gamma_l^*\})$, we give a closed-form lower bound on the competitive ratio $\mathtt{P}_1(\{\gamma_l^*\})$ in the following corollary:

Corollary 1. *(Closed-Form Lower Bound on the Competitive Ratio) The competitive ratio of OWA is lower bounded by* $\alpha' = \max\{1/L, c\} \cdot (1 - K^{-\frac{1}{2}})$, *where* $c = \min_{l\in[L]} \underline{u}_l / \max_{l\in[L]} \overline{u}_l$ *is a constant derived from the problem instance.*

Proof. (Sketch of Proof) To prove Corollary 1, we find a feasible solution to the optimization problem $\mathtt{P}_1$ and show that the competitive ratio achieved by this solution in GMPMW is no less than α'. Then, since $\alpha = \mathtt{P}_1(\{\gamma_l^*\})$ is the optimal value obtained on problem $\mathtt{P}_1$ by the optimal solution γ_l^*, it is lower bounded by α'. The complete proof is given in Appendix E.4. □

Note that the numerical value of α consistently exceeds its lower bound α'. We provide numerical results in Appendix F to illustrate the actual behavior of the competitive ratio α. When there is only one worker (i.e., $L = 1$), the competitive ratio α is lower bounded by $\alpha' = 1 - K^{-\frac{1}{2}}$, matching the previous best result on the original single-worker GMP [Alaei et al., 2013]. This result confirms that our algorithm design is consistent with the previous best algorithm design for the single-worker GMP, while also addressing the multiple workers in GMPMW.

5.2 TIGHTNESS OF THE COMPETITIVE RATIO

Now that we have obtained the competitive ratio of the OWA algorithm, we investigate how tight this performance bound is. First, we give a formal definition of the tightness of the competitive ratio of an online algorithm in Definition 2.

Definition 2. *(Tightness) The competitive ratio* α *for an online algorithm ALG is tight if and only if there exists a task sequence I where* $\mathbb{E}_{i\sim I}[ALG(i)] = \alpha \max_{i\in\Omega^I}[OPT(i)]$.

Next, we prove that the competitive ratio $\alpha = \mathtt{P}_1(\{\gamma_l^*\})$ achieved by the OWA algorithm on GMPMW is tight.

Theorem 5. *(Tightness of* α*) For every combination of* K *and* L*, there exists a tasks sequence* I*, where the performance ratio achieved by the OWA algorithm is exactly* $\mathbb{E}_{i\sim I}[ALG(i)]/\max_{i\in\Omega^I}[OPT(i)] = \alpha = \mathtt{P}_1(\{\gamma_l^*\})$.

Proof. We focus on the task sequence where the reward lower and upper bounds for each l are $\underline{u}_l = 0$ and $\overline{u}_l = u$. In this case, the competitive ratio of OWA is $\alpha = \mathtt{P}_1(\{\gamma_l^*\}) = \min_l \gamma_l^*$. To induce the worst-case performance, the adversary will arrange the task sequence in the following way: The reward for processing each task t by the worker $l' = \arg\min_l \gamma_l^*$ is set to $u_{t,l'} = u$, and $u_{t,l} = 0$ for processing each task by every other worker l. The offline optimal result is thus $\max \mathrm{OPT}(i) = Tu$, which is obtained when the offline algorithm processes each task t by worker l', reached when $\sum_{t\in[T]} r_{t,l'} < K$. In this case, the expected total reward obtained by the OWA algorithm is exactly $\gamma_{l'}^* Tu$, resulting in a performance ratio of $\gamma_{l'}^* = \min_l \gamma_l^* = \mathtt{P}_1(\{\gamma_l^*\})$. □

5.3 OPTIMALITY OF THE OWA ALGORITHM

We investigate whether any online algorithm can achieve a higher competitive ratio on GMPMW than that of OWA. For the single-worker GMP, Alaei et al. [2013] proved the existence of asymptotically optimal solutions only for the special case where the reward lower bound is 0. In what follows, we show that the competitive ratio α of OWA is also asymptotically optimal with respect to K when the reward lower bound $\underline{u}_l$ for every worker l is 0 (i.e., $c = 0$), such that OWA recovers the best existing solution for GMP [Alaei et al., 2013] when there is only one worker (i.e., $L = 1$). However, since α does not have a closed-form expression, we will first prove that its lower bound α' is asymptotically optimal, as stated in Theorem 6.

Theorem 6. *(Asymptotic Optimality of* α'*) When the reward lower bound of each worker is* 0*,* α' *is an asymptotically optimal competitive ratio for the OWA algorithm on GMPMW, meaning that* α' *approaches the maximum possible competitive ratio when budget* K *approaches infinity.*

Proof. (Sketch of Proof) To prove Theorem 6, we show that there exists an adversary for which no online algorithm can achieve a competitive ratio higher than $\alpha^* = \frac{1}{L}$. The complete proof is in Appendix E.5. □

Remark 1. *Since Corollary 1 establishes that* $\alpha' \leq \alpha$*, Theorem 6 directly implies that, when* $c = 0$*, the competitive ratio* α *of OWA also approaches the maximum competitive ratio as the budget* K *approaches infinity.*

Note that the asymptotic optimality of [Alaei et al., 2013] was also proved in the case where the reward lower bound is 0, such that our result implies that OWA is a natural

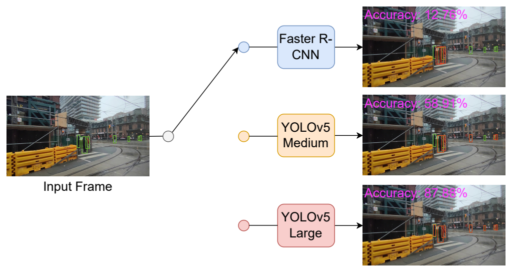


Figure 2: Video analytics and accuracy calculation.

generalization of the existing best solution to incorporate the multiple workers in GMPMW. In Appendix G, we further provide numerical results to study the impact of various system parameters on the competitive ratio of OWA.

6 EVALUATION OF OWA BY CASE STUDY

To further evaluate the performance of OWA, we conduct trace-driven experiments under a practical system of real-time video analytics at the Internet edge with multiple deployed machine learning models.

6.1 MODEL SELECTION FOR REAL-TIME VIDEO ANALYTICS

Real-time video analysis requires sophisticated machine learning models, but the edge devices, such as tablets and laptops, are equipped with limited batteries. In many situations, the edge devices are not connected to a persistent power supply, so it is important to efficiently manage the workload on the edge devices. Here we consider a general scenario where an edge computing device, equipped with multiple machine learning models, is used for real-time video analytics, processing a sequence of video chunks in practical applications such as traffic monitoring in smart cities and hazard prevention. Different machine learning models (workers) generate different accuracy values (reward) and consume different amounts of energy (resource). We need to decide whether to process or discard a video chunk and, if so, which machine learning model to use, in order to maximize the overall accuracy within the energy constraint.

This application scenario of real-time video analysis with multiple models deployed on the edge device is consistent with the problem formulation of the GMPMW. The video chunks are the incoming tasks, and the multiple machine learning models to process each video chunk are the multiple workers in the GMPMW. The probability distribution of the energy consumption of each model $\mathcal{R}_{t,l}$ can be estimated by model profiling and video pre-processing Hung et al. [2018], and the upper bound $\overline{u}_l$ and lower bound $\underline{u}_l$ can also be obtained by profiling Zhang et al. [2017]. The edge device does not know the accuracy and energy consumption of processing each video chunk before that chunk arrives, and only after processing a chunk using a model can it know the corresponding accuracy and energy consumption. The energy budget of the edge device is the limited resource budget.

6.2 VIDEO TRACE COLLECTION AND MODEL PROFILING

For the video traces, we use a Xiaomi 12 Pro Android smartphone equipped with a Sony IMX766 photosensor to capture the traffic on the road as well as the network condition. We collect 4 sets of video traces with different lengths, labeled as Trace 1–4. The video frames are grouped into video chunks, with each video chunk containing the video frames in 3 seconds. The video chunks are then sent to the edge device for analysis, so each video chunk is a task.

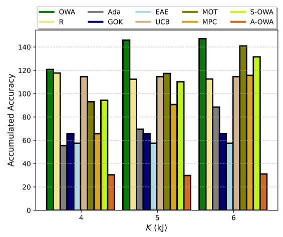
We deploy 3 machine learning models on a laptop computer powered by an Intel Core i5-11320H CPU with integrated graphics to analyze the video chunks. The machine learning models deployed are Faster R-CNN Ren et al. [2017], YOLOv5 Jocher [2020] with medium backbone, and YOLOv5 with large backbone. To obtain the power consumption of each model, we use the Python implementation of Intel's Running Average Power Limit (pyRAPL) Spirals [2023]. The video frames are grouped into video chunks, each containing 3 seconds of video frames. The video chunks are then sent to the edge device for analysis, with each video chunk being a task.

We obtain the accuracy of each model by model profiling Zhang et al. [2017], and we adopt an accuracy criterion based on the Intersection over Union metric, which is widely accepted in video analytics Lin et al. [2014], Redmon et al. [2016], Ren et al. [2017]. In Figure 2, each inference generated by an object detection model yields a set of predicted bounding boxes (red boxes). In parallel, we construct a set of ground truth objects, each represented by its own bounding box (green boxes). These ground truth bounding boxes are derived from a highly reliable model, the Faster R-CNN Ren et al. [2017], which uses the Resnet 50 He et al. [2016] backbone. The accuracy of our model is determined by comparing these two sets of bounding boxes.

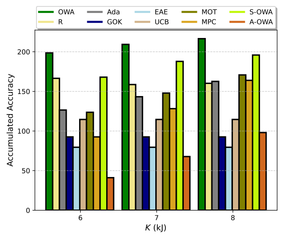
We profile the accuracy and the energy consumption of each model in the same environment. The accuracy and energy measurements are executed once per second and averaged over each 3-second video chunk. Further details are given in Appendix H.

6.3 BENCHMARKS

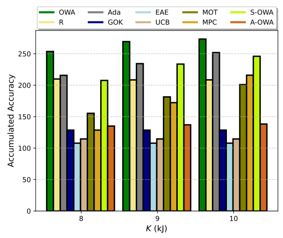
We compare the performance of OWA against that of the following benchmarks (with further details on each of them



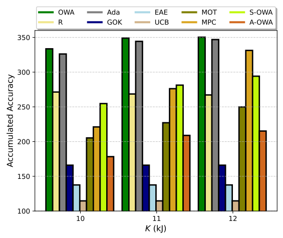
(a) Trace 1 (1200s).



(b) Trace 2 (1800s).



(c) Trace 3 (2400s).



(d) Trace 4 (3000s).

Figure 3: Comparing OWA against benchmarks.

given in Appendix H): (i) R: the Random algorithm randomly decides whether to process a video chunk with a model or to discard it; (ii) Ada: the Adaptive algorithm selects a random model to process a new chunk if it has previously discarded any chunk; (iii) GOK: the Greedy Online Knapsack algorithm processes every video chunk using the model with the highest efficiency; (iv) EAE: Exploration and Exploitation [Audibert et al., 2009]; (v) UCB: Upper Confidence Bound Bandit [Garivier and Moulines, 2011]; (vi) MOT: Multi-worker One-way Trading [Cao et al., 2020]; (vii) MPC: Model Predictive Control [Morari and Lee, 1999]; (viii) S-OWA: Single-Worker OWA implements a single-worker version of OWA; (ix) A-OWA: Average OWA first selects a fixed model and then decides whether to process the video chunks. Further details on each of the benchmarks are given in Appendix H.

6.4 PERFORMANCE

To compare the performance of the OWA algorithm against that of the benchmarks, we apply all algorithms to each video trace with different resource budgets K, as shown in Figure 3.

In all settings, we observe that the OWA algorithm outperforms all benchmarks under all conditions on our video traces. We also have some observations on the performance of each of the benchmarks. We will discuss them in three groups: ① Random, Adaptive, and GOK; ② EAE, UCB, MOT, and MPC; and ③ S-OWA and A-OWA.

The Random algorithm performs worse than OWA because it does not consider the resource constraints. The adaptive algorithm considers the resource constraint, but simply controls the number of tasks (chunks) processed, rather than the resource consumption, so it performs worse than OWA. The GOK algorithm sticks to the worker (model) with the highest average efficiency, but the most efficient worker may not fully utilize all of the resource (energy), leading to inferior performance.

EAE and UCB perform worse than OWA because they are not aware of the resource constraint. MOT performs worse than the OWA algorithm because, in our system, the reward (accuracy) and the resource consumption are not known when a video chunk arrives. However, these are important variables for the decision-making process in the MOT algorithm. When MOT can only use the profiled data to make decisions, its performance suffers. MPC performs worse than the OWA algorithm because the real-world street scenes captured in our video traces are highly fluctuating. As a result, the predictions made by MPC are not accurate.

Finally, we discuss S-OWA and A-OWA, which are different variants of OWA. S-OWA focuses only on the best possible worker (model) according to model profiling. It performs worse than OWA because it does not balance between the reward and resource consumption by utilizing all workers. On the other hand, A-OWA first chooses the worker (model) and then makes a decision based on that worker's baseline. This strategy of A-OWA leads to worse performance than OWA because each worker only updates its own baseline, but the different reward and resource consumption levels among different workers are coupled in GMPMW. In contrast, OWA updates the baselines in a joint manner and assigns each task to different workers based on the model profiling result, thus achieving superior performance.

In conclusion, our experimental results demonstrate the excellent capability of the OWA algorithm to utilize the multiple workers in GMPMW in a variety of realistic system settings, and they show the importance of properly handling the multiple workers in the proposed approach.

7 CONCLUSION

In this paper, we study GMPMW and consider the complicated trade-off between reward accumulation and resource consumption imposed by multiple workers. We propose the OWA algorithm to tackle the new challenges. OWA balances the resource consumption across workers through a set of assignment guarantees and tracks the virtual resource consumption to capture the long-term resource availability. The competitive ratio of OWA, $\alpha = \mathtt{P}_1(\{\gamma_l^*\})$, which is lower bounded by $\alpha' = \max\{1/L, c\} \cdot (1 - K^{-\frac{1}{2}})$, is tight and approaches the maximum competitive ratio. When there is only one worker, this lower bound (thus the competitive ratio α) matches the best existing result for the original GMP. We perform trace-driven experiments to demonstrate the excellent capability of the OWA algorithm to accommodate multiple workers in GMPMW.

References

Saeed Alaei. Bayesian Combinatorial Auctions: Expanding Single Buyer Mechanisms to Many Buyers. *SIAM Journal on Computing*, 43(2):930–972, 2014.

Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The Online Stochastic Generalized Assignment Problem. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 11–25, 2013.

Ayoub Amil, Ali Makhdoumi, and Yehua Wei. Multi-Item Order Fulfillment Revisited: LP Formulation and Prophet Inequality. *Management Science*, 0(0):1–19, 2025.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with Knapsacks. *Journal of the ACM*, 65(3):1–55, 2018.

Hans-Joachim Böckenhauer, Dennis Komm, Richard Královič, and Peter Rossmanith. The online knapsack problem: Advice and randomization. *Theoretical Computer Science*, 527:61–72, 2014.

Shant Boodaghians, Federico Fusco, Philip Lazos, and Stefano Leonardi. Pandora's Box Problem with Order Constraints. In *Proceedings of the ACM Conference on Economics and Computation*, pages 439–458, 2020.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Niv Buchbinder, Joseph Seffi Naor, et al. The Design of Competitive Online Algorithms via a Primal–Dual Approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.

Ying Cao, Bo Sun, and Danny HK Tsang. Optimal Online Algorithms for One-Way Trading and Online Knapsack Problems: A Unified Competitive Analysis. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1064–1069, 2020.

Shuchi Chawla, Jason D Hartline, David L Malec, and Balasubramanian Sivan. Multi-parameter Mechanism Design and Sequential Posted Pricing. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 311–320, 2010.

Brian C Dean, Michel X Goemans, and Jan Vondrák. Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

John P Dickerson, Karthik A Sankararaman, Aravind Srinivasan, and Pan Xu. Allocation Problems in Ride-sharing Platforms: Online Matching with Offline Reusable Resources. *ACM Transactions on Economics and Computation*, 9(3):1–17, 2021.

Davide Drago, Andrea Celli, and Marek Elias. Bandits with Knapsacks and Predictions. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 1189–1206, 2024.

Ran El-Yaniv, Amos Fiat, Richard M Karp, and Gordon Turpin. Optimal Search and One-Way Trading Online Algorithms. *Algorithmica*, 30:101–139, 2001.

Hossein Esfandiari, MohammadTaghi HajiAghayi, Brendan Lucier, and Michael Mitzenmacher. Online Pandora's Boxes and Bandits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1885–1892, 2019.

Moran Feldman, Ola Svensson, and Rico Zenklusen. Online Contention Resolution Schemes with Applications to Bayesian Selection Problems. *SIAM Journal on Computing*, 50(2):255–300, 2021.

Aurélien Garivier and Eric Moulines. On Upper-Confidence Bound Policies for Switching Bandit Problems. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 174–188, 2011.

Khashayar Gatmiry, Thomas Kesselheim, Sahil Singla, and Yifan Wang. Bandit Algorithms for Prophet Inequality and Pandora's Box. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 462–500, 2024.

Anupam Gupta and Viswanath Nagarajan. A Stochastic Probing Problem with Applications. In *Proceedings of the International Conference on Integer Programming and Combinatorial Optimization*, pages 205–216, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. VideoEdge: Processing Camera Streams using Hierarchical Clusters. In *Proceedings of the IEEE/ACM Symposium on Edge Computing*, pages 115–131, 2018.

Nicole Immorlica, Karthik Sankararaman, Robert Schapire, and Aleksandrs Slivkins. Adversarial Bandits with Knapsacks. *Journal of the ACM*, 69(6):1–47, 2022.

Hongyi Jiang and Samitha Samaranayake. Approximation Algorithms for Capacitated Assignment with Budget Constraints and Applications in Transportation Systems. In *Proceedings of the International Computing and Combinatorics Conference*, pages 94–105, 2022.

Jiashuo Jiang, Will Ma, and Jiawei Zhang. Tight Guarantees for Multi-unit Prophet Inequalities and Online Stochastic Knapsack. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1221–1246, 2022.

Glenn Jocher. Yolov5 by ultralytics. `https://github.com/ultralytics/yolov5`, 2020. Accessed: 2024-01-15.

Zihao Li, Hao Wang, and Zhenzhen Yan. Sample-Based Online Generalized Assignment Problem with Unknown Poisson Arrivals, 2023.

Qiulin Lin, Hanling Yi, John Pang, Minghua Chen, Adam Wierman, Michael Honig, and Yuanzhang Xiao. Competitive Online Optimization under Inventory Constraints. *SIGMETRICS Perform. Eval. Rev.*, 47(1):35–36, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755, 2014.

Haodong Liu, Huili Zhang, Kelin Luo, Yao Xu, Yinfeng Xu, and Weitian Tong. Online generalized assignment problem with historical information. *Computers & Operations Research*, 149:106047, 2023.

Aranyak Mehta et al. Online Matching and Ad Allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

Jason D Papastavrou, Srikanth Rajagopalan, and Anton J Kleywegt. The Dynamic and Stochastic Knapsack Problem with Deadlines. *Management Science*, 42(12):1706–1718, 1996.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

Spirals. pyrapl. `https://pypi.org/project/pyRAPL/`, 2023. Accessed: 2024-01-15.

Aravind Srinivasan and Pan Xu. The Generalized Magician Problem under Unknown Distributions and Related Applications. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1219–1227, 2022.

Ruoyu Wu, Wei Bao, and Liming Ge. Online Task Assignment with Controllable Processing Time. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 5466–5474, 2023.

Toru Yoshinaga and Yasushi Kawase. Size-stochastic knapsack online contention resolution schemes, 2023.

Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, pages 377–392, 2017.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. Budget Constrained Bidding in Keyword Auctions and Online Knapsack Problems. In *Proceedings of the International Conference on World Wide Web*, page 1243–1244, 2008.

Online Generalized Magician's Problem with Multiple Workers (Supplementary Material)

Ruoyu Wu[1] **Wei Bao**[1] **Ben Liang**[2] **Liming Ge**[1]

[1]School of Computer Science, The University of Sydney, Sydney, NSW, Australia
[2]Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada

A REAL-WORLD APPLICATIONS OF GMPMW

GMPMW has real-world applications in various areas. The following are a few examples.

Cloud Computing Task Processing A cloud provider offers a machine learning (ML) inference service. It receives ML task requests, where different ML models produce different accuracies and require different amounts of energy. The cloud provider needs to decide which task to accept or reject, and if so, which model to use to process each task, in order to maximize the accumulated accuracy. In the experiment section of the paper (and Appendix H), this scenario is used as a case study to evaluate our proposed algorithm.

Labor Outsourcing A labor outsourcing company receives job requests, and different workers can be assigned to the job. Different workers generate different profits and require different hourly wages, while the total wage is budgeted. The company decides which job to accept or reject, and which worker to assign to which task, in order to maximize the total profit.

Online Ad Placement An online ad service provider receives ad placement requests, and different types of ads can be placed for each request. Different types of ads generate different interaction rates and consume different amounts of ad time, while the total amount of ad time is budgeted. The service provider decides which request to accept, and if so, which type of ad to serve, in order to maximize the overall interaction rate. In this scenario, the maximum total ad time is the limited resource, and the different types of ads are different workers.

B COMPREHENSIVE COMPARISON OF GMP AND GMPMW WITH RELATED PROBLEMS

A distinct feature of GMP and GMPMW is their unique progression of task cost observability, which refers to the key instants in online decision-making when the observability of task costs improves, as well as the extent of those improvements. This uniqueness in GMP and GMPMW stems from their complete progression of task cost observability—initially unknown, partially observable as a probability distribution upon task arrival, and fully revealed as an exact value upon task completion, involving all three levels of task cost observability.

To further clarify this distinction, Figure 4 compares the progression of task cost observability across different problems. Task cost observability may increase at four key instants: before task arrival, upon task arrival, upon discarding, and after processing, categorized into three levels: Unknown (no information), Distribution (probability distribution known), and Value (exact value known). We compare GMP and GMPMW (red) against the Online Generalized Assignment Problem (OGAP), Online Stochastic Generalized Assignment Problem (OSGAP), Online Knapsack Problem (OKP), Online Stochastic Knapsack Problem (OSKP), and the Bandit with Knapsack (BwK) problem, including its Full Feedback (BwK-F) and Bandit Feedback (BwK-B) variants. Unlike other problems that experience only two levels of task cost observability, GMP and GMPMW are the only ones that undergo the full progression involving all three levels. Figure 1 summarizes Tables 1–3,

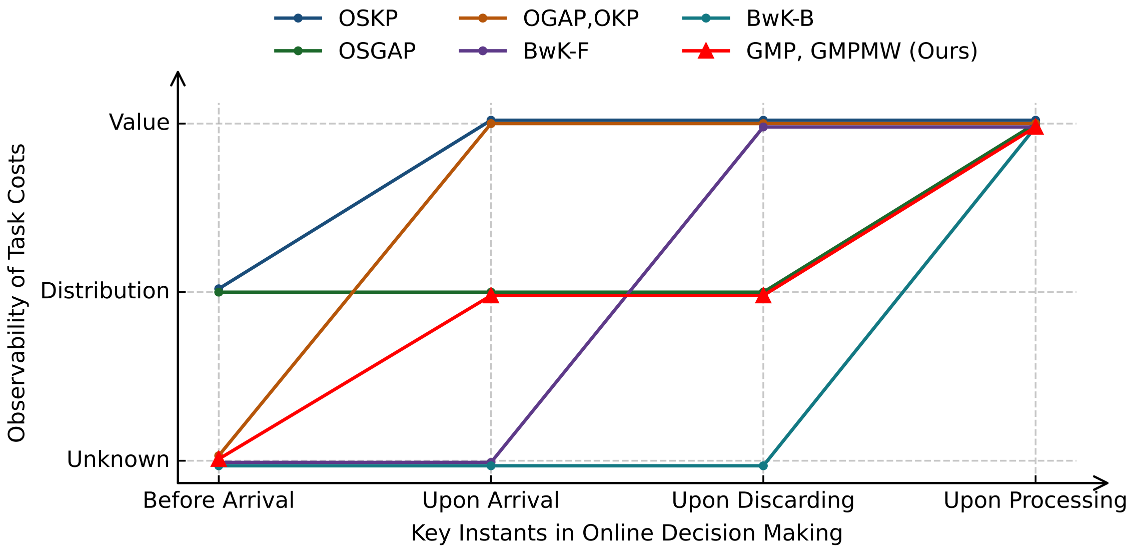


Figure 4: (Reproduced from Figure 1) Progression of task cost observability in online problems.

Table 1: Task Observability of Rewards (R) and Costs (C) in GMP, OGAP, and OSGAP

Online Problems	Before Arrival	Upon Arrival	Upon Discarding	Upon Processing	Typical Work
GMP and GMPMW	R: Unknown. C: Unknown.	R: Unknown. C: Distribution.	R: Unknown. C: Distribution.	R: Value. C: Value.	[Alaei et al., 2013, Alaei, 2014]
OGAP	R: Unknown. C: Unknown.	R: Value. C: Value.	R: Value. C: Value.	R: Value. C: Value.	[Liu et al., 2023, Li et al., 2023]
OSGAP	R: Distribution. C: Distribution.	R: Value. C: Distribution.	R: Value. C: Distribution.	R: Value. C: Value.	[Alaei et al., 2013, Yoshinaga and Kawase, 2023]

where we compare the progression of observability of both task reward and cost across different problems. For conciseness, we use "R" to denote the observability of task reward and "C" to denote the observability of task cost throughout Tables 1–3. The details of this comparison, along with the tables, are discussed in the remainder of this section.

B.1 GMP, OGAP, AND OSGAP

In this section, we first introduce the origins and history of GMP, followed by an introduction to OGAP and OSGAP. We will also compare the observability of task reward and cost in these problems, which is summarized in Table 1.

In GMP, which was originally introduced by Alaei et al. [2013], a decision maker needs to decide which task to process in a task sequence, in order to maximize the cumulative reward within the resource budget K. Alaei et al. [2013] studied GMP and achieved a competitive ratio of $1 - K^{-1/2}$. In GMP, task reward is unobservable: The decision-maker has no knowledge about task reward before or upon task arrival, and the task reward is revealed only after processing. Task cost has progressively improved observability: The decision maker has no knowledge about task cost before task arrival, then knows the distribution of task cost upon arrival, and knows the value of task cost only after processing it. A special case of GMP with random 0-1 cost is the Magician's Problem (MP), which was then studied by Alaei [2014] with a competitive ratio of $1 - (K + 3)^{-1/2}$ achieved. Recently, Srinivasan and Xu [2022] studied a variant of GMP considering scenarios with unknown distributions of resource consumption. Owing to the large-scale setting and the unique observability of tasks, GMP has found widespread application in various fields, including e-commerce [Amil et al., 2025] and transportation [Jiang and Samaranayake, 2022]. However, none of the above works can accommodate the multiple workers in GMPMW.

Table 2: Task Observability of Rewards (R) and Costs (C) in OKP, OSKP, and BwK

Online Problems	Before Arrival	Upon Arrival	Upon Discarding	Upon Processing	Typical Work
OKP	R: Unknown. C: Unknown.	R: Value. C: Value.	R: Value. C: Value.	R: Value. C: Value.	[Zhou et al., 2008, Cao et al., 2020]
OSKP (Stochastic Reward and Cost)	R: Distribution. C: Distribution.	R: Value. C: Value.	R: Value. C: Value.	R: Value. C: Value.	[Papastavrou et al., 1996]
OSKP (Stochastic Reward)	R: Distribution. C: Value.	R: Value. C: Value.	R: Value. C: Value.	R: Value. C: Value.	[Jiang et al., 2022]
OSKP (Stochastic Cost)	R: Value. C: Distribution.	R: Value. C: Value.	R: Value. C: Value.	R: Value. C: Value.	[Dean et al., 2008]
BwK-B	R: Unknown. C: Unknown.	R: Unknown. C: Unknown.	R: Unknown. C: Unknown.	R: Value. C: Value.	[Badanidiyuru et al., 2018, Immorlica et al., 2022]
BwK-F	R: Unknown. C: Unknown	R: Unknown. C: Unknown.	R: Value. C: Value.	R: Value. C: Value.	[Badanidiyuru et al., 2018, Immorlica et al., 2022]

The unique task cost observability and the large-scale setting also enabled GMP to be adopted to tackle other online problems, such as OSGAP [Alaei et al., 2013, Alaei, 2014], which is also studied in the large-scale setting. In OSGAP, each task belongs to a type drawn from a known distribution. A task's type determines its reward and cost distribution, both known in advance. Therefore, by knowing the distribution of task types, the decision-maker also knows the distributions of both task reward and cost. Upon arrival, the task's type is revealed, specifying the reward and the cost distribution. Recently, Yoshinaga and Kawase [2023] extended OSGAP to consider a more limited resource budget while maintaining the large-scale setting. Liu et al. [2023] and Li et al. [2023] studied OGAP as an adversarial variant of OSGAP, where the reward and cost of each task are arranged by an adversary and are revealed only upon arrival. However, as shown in Figure 4 and Table 1, despite the differing initial knowledge of task costs in OSGAP (distribution) and OGAP (none), both problems follow a similar observability progression: the exact task cost is fully revealed immediately upon arrival (OGAP) or upon processing (OSGAP), without an intermediate level. Consequently, none of these problems considers the full progression of task cost observability involving all three levels as in GMP and GMPMW.

B.2 OKP, OSKP, AND BWK

In this section, we introduce OKP, OSKP, and BwK, and discuss the task observability in these problems, which is summarized in Table 2. In OKP [Zhou et al., 2008, Cao et al., 2020], the decision-maker also chooses from a sequence of tasks to process to maximize the reward accumulation within the cost budget. OKP [Zhou et al., 2008, Böckenhauer et al., 2014] assumes that the decision-maker has no information on the reward and cost of tasks except the bounded ratio between task reward and cost, and the reward and cost of a task are fully revealed upon task arrival. A stochastic variant of OKP is the Online Stochastic Knapsack Problem (OSKP). In OSKP, task costs are either deterministic (with stochastic rewards) or stochastic (with deterministic or stochastic rewards). In the deterministic cost setting [Jiang et al., 2022], costs are known by the decision maker at the beginning, while in the stochastic cost setting [Papastavrou et al., 1996, Dean et al., 2008], the cost of each task follows a known distribution, with its exact value revealed upon task arrival. However, as shown in Figure 4 and Table 2, despite the differing initial knowledge of task costs in the stochastic-cost OSKP (distribution) and OKP (none), both problems follow the same observability progression: the exact task cost is fully revealed immediately upon arrival, without an intermediate level. Moreover, as shown in Table 2, in the stochastic-reward OSKP, when task costs are deterministic, the decision-maker knows the task costs even before task arrival. Consequently, none of these problems considers the full progression of task cost observability involving all three levels as in GMP and GMPMW.

Table 3: Task Observability of Rewards (R) and Costs (C) in OMuPI, COPM, and the Stochastic Probing Problem

Online Problems	Before Arrival	Upon Arrival	Upon Discarding	Upon Processing	Typical Work
OMuPI	R: Distribution. C: Value.	R: Value. C: Value.	R: Value. C: Value.	R: Value. C: Value.	[Feldman et al., 2021, Jiang et al., 2022]
COPM	R: Distribution. C: Value.	R: Distribution. C: Value.	R: Distribution. C: Value.	R: Value. C: Value.	[Chawla et al., 2010, Feldman et al., 2021]
Stochastic Probing	R: Distribution. C: Value.	R: Distribution. C: Value.	R: Distribution. C: Value.	R: Value. C: Value.	[Gupta and Nagarajan, 2013, Feldman et al., 2021]

In BwK [Badanidiyuru et al., 2018, Immorlica et al., 2022, Drago et al., 2024], the decision-maker does not know a task's reward or cost even upon its arrival. Two feedback settings are considered: bandit feedback (BwK-B), where both values are revealed only after processing the task, and full feedback (BwK-F), where they are revealed after the task is either processed or discarded. However, as shown in Figure 4 and Table 2, both BwK-B and BwK-F start with no knowledge of task costs and follow a similar observability progression, where the exact task cost is directly revealed after the task is discarded (BwK-B) or processed (BwK-F). Consequently, none of these problems considers the full progression of task cost observability involving all three levels as in GMP and GMPMW.

B.3 THE ONLINE CONTENTION RESOLUTION SCHEME

Recently, Feldman et al. [2021] introduced the Online Contention Resolution Scheme (OCRS) as a rounding scheme for solving online submodular function optimization problems. The core of OCRS involves generating a fractional solution for a linear relaxation of the problem using available prior knowledge, followed by a stochastic and sequential rounding process to derive a feasible integer solution for the original online problem. OCRS has been found capable of solving the Online Bayesian Optimization Problem (BOP), including the Online Multi-Unit Prophet Inequality (OMuPI) [Feldman et al., 2021, Jiang et al., 2022], the Constrained Oblivious Posted Price Mechanisms Problem (COPM) [Chawla et al., 2010, Feldman et al., 2021], and the Stochastic Probing Problem [Gupta and Nagarajan, 2013, Feldman et al., 2021]. In OMuPI, task rewards follow known distributions and are revealed upon arrival, with task costs known to the decision-maker at the beginning. In both COPM and the Stochastic Probing Problem, task rewards follow known distributions and are revealed only after processing, with task costs known to the decision-maker at the beginning. Later, Jiang et al. [2022] adopted OCRS to solve a deterministic-cost OSKP. However, as shown in Figure 4 and Tables 2–3, all of these works consider deterministic task costs known to the decision-maker at the beginning. Moreover, the absence of prior knowledge about task rewards and costs distinguishes GMP and GMPMW from BOP, making it impossible to construct a meaningful linear relaxation in advance for GMP. Consequently, GMP and GMPMW require fundamentally different approaches from OCRS.

B.4 OTHER ONLINE OPTIMIZATION PROBLEMS

Further to Section 2, here we summarize other online optimization problems that are less relevant to GMP and GMPMW. The One-Way Trading Problem (OTP) involves continuous resource consumption with task reward arriving in an adversarial order, considering an infinite time horizon and resource consumption controlled by the decision maker [Cao et al., 2020, El-Yaniv et al., 2001, Lin et al., 2019]. Compared with the OTP, GMP and GMPMW are distinct in their unknown and limited time horizon and uncertain resource consumption. The Online Bipartite Matching Problem deals with the matching of online tasks and offline machines, where the time horizon is known to the decision maker [Mehta et al., 2013, Dickerson et al., 2021, Wu et al., 2023]. The online Pandora's Box problem introduces a unique approach where the decision maker strategically reveals item values, in order to maximize the maximum value (rather than the sum value) of revealed items minus the cost of revealing, with no limit on total cost, controlled item arrival order, and known item number [Boodaghians et al., 2020, Esfandiari et al., 2019, Gatmiry et al., 2024]. These problems are structurally different from GMP and GMPMW.

C PROOF OF THEOREM 1

(Convex Equivalence) There exists a convex optimization problem $\mathtt{P}'$, such that its optimal solution is also the optimal solution to $\mathtt{P}_1$.

Proof. Recall that the optimization problem $\mathtt{P}_1$ is defined as

$$\mathtt{P}_1 \quad \max_{\gamma_l, \forall l} \quad \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma_{l'} \underline{u}_{l'}}{\bar{u}_l} + \gamma_l \tag{17}$$

$$\textbf{s.t.} \quad K \geq \frac{1}{(1 - \sum_{l \in [L]} \gamma_l)(1 - \max_l \gamma_l L)}, \tag{18}$$

$$\max_l \gamma_l < \frac{1}{L}, \tag{19}$$

$$\gamma_l \geq 0, \ \forall l. \tag{20}$$

(i) Note that the objective function in Eq. (17) is equivalent to

$$\sum_{l' \in [L]} \gamma_{l'} - \sum_{l' \in [L] \setminus l} \gamma_{l'} (1 - \frac{\underline{u}_{l'}}{\bar{u}_l}). \tag{21}$$

In addition, by Eq. (19), Eq. (18) is equivalent to

$$\sum_{l \in [L]} \gamma_l \leq 1 - \frac{1}{K - \max_l \gamma_l K L}. \tag{22}$$

Therefore, $\mathtt{P}_1$ is equivalent to

$$\mathtt{P}_2 \quad \max_{\gamma_l, \forall l} \quad \min_l \sum_{l' \in [L]} \gamma_{l'} - \sum_{l' \in [L] \setminus l} \gamma_{l'} (1 - \frac{\underline{u}_{l'}}{\bar{u}_l}) \tag{23}$$

$$\textbf{s.t.} \quad \sum_{l \in [L]} \gamma_l \leq 1 - \frac{1}{K - \max_l \gamma_l K L}, \tag{24}$$

$$\max_l \gamma_l < \frac{1}{L}, \tag{25}$$

$$\gamma_l \geq 0, \ \forall l. \tag{26}$$

(ii) There exists an optimal solution that satisfies the equality of Eq. (24). Otherwise, for an optimal solution not satisfying the equality of Eq. (24), we can increase the γ_l that is smaller than $1/L$ until the equality of Eq. (24) is satisfied. During this process, Eq. (23) will not decrease.

(iii) Therefore, an optimal solution of $\mathtt{P}_1$ can be obtained by optimally solving

$$\mathtt{P}_3 \quad \max_{\gamma_l, \forall l} \quad \min_l 1 - \frac{1}{K - \max_l \gamma_l K L} - \sum_{l' \in [L] \setminus l} \gamma_{l'} (1 - \frac{\underline{u}_{l'}}{\bar{u}_l}) \tag{27}$$

$$\textbf{s.t.} \quad \max_l \gamma_l < \frac{1}{L}, \tag{28}$$

$$\gamma_l \geq 0, \ \forall l. \tag{29}$$

(iv) We set $w = \max_l \gamma_l$, then $\mathtt{P}_3$ is equivalent to

$$\mathtt{P}' \quad \max_{w, \{\gamma_l\}_{l \in [L]}} \quad \min_l 1 - \frac{1}{K - wKL} - \sum_{l' \in [L] \setminus l} \gamma_{l'} (1 - \frac{\underline{u}_{l'}}{\bar{u}_l}) \tag{30}$$

$$\textbf{s.t.} \quad 0 \leq w < \frac{1}{L}, \tag{31}$$

$$0 \leq \gamma_l \leq w, \, \forall l. \tag{32}$$

(v) In Eq. (30), the second term $1/(K - wKL)$ is convex when $w < 1/L$, the last term is linear, so the objective function $1 - \frac{1}{K-wKL} - \sum_{l' \in [L] \setminus l} \gamma_{l'}(1 - \frac{u_{l'}}{\bar{u}_l})$ is concave, the minimization of this function is concave, and Eq. (30) is convex. With Eq. (31) and Eq. (32), $\mathtt{P}'$ is a convex optimization problem.

(vi) Therefore, an optimal solution to the convex optimization problem $\mathtt{P}'$ is also an optimal solution to $\mathtt{P}_1$. □

D OWA ALGORITHM COMPLEXITY ANALYSIS

Complexity of the Pre-Calculation Phase. Before the first task arrives, we need to solve the optimization problem $\mathtt{P}_1$ to obtain the parameters γ_l^*. Since, by Theorem 1 and its proof (Appendix C), the optimal solution to $\mathtt{P}_1$ can be obtained by solving the convex optimization problem $\mathtt{P}'$, which is given in Eq. (29)–(31). We can solve $\mathtt{P}'$ by standard solvers for convex optimization problems, such as the interior-point method (or the barrier method). Specifically, if we solve $\mathtt{P}'$ using the *barrier method* [Boyd and Vandenberghe, 2004] with the starting point as $w = 0$ and $\gamma_l = 0$ for each l, it will take at most $\lceil 2\log_2(\frac{5\mathtt{P}_1(\{\gamma_l^*\})}{\epsilon})\rceil(\frac{1}{2p} + c)$ Newton steps (centering steps), where ϵ is the error tolerance, $\frac{1}{p} = \frac{20-8\alpha}{\alpha\beta(1-2\alpha)^2}$ where α and β are the backtracking parameters, and $c = \log_2(\log_2(1/\epsilon))$. Each Newton step has a complexity of $\mathcal{O}(L^3)$. Since the pre-calculation phase is performed offline before the online process, this complexity can be well accepted.

Complexity of the Worker-Assignment Phase and the Processing Phase. In the worker-assignment phase, the loops on lines 5–7 result in a complexity of $\mathcal{O}(1)$. Then we go through the processing phase, where the update of Θ_{t+1} on lines 9–16 has $\mathcal{O}(1)$ complexity. Therefore, the total complexity of the worker-assignment phase and the processing phase is $\mathcal{O}(1)$.

Complexity of the Baseline-Calibration Phase. In the baseline-calibration phase, we need to calculate the resource utilization function h_{t+1} according to Eq. (12). A straightforward solution to perform the convolution operation is to calculate its value directly by performing an integral, which can have high computational complexity. However, as we will see later in the proof of Theorem 2, every $h_t(w)$ has a finite length (i.e., $h_t(w) = 0$ if $w < 0$ or $w \geq K$). Therefore, we may use Fast Fourier Transform (FFT) to efficiently compute the convolution in this step. Let D be the number of discrete-time samples of each continuous function $\overline{h}_t$ and $\overline{g}_t$. **First**, for each task t, we update h_{t+1} by Eq. (12) in Line 22. To calculate each $\overline{g}_t$ by Eq. (14), we first calculate each $(1/\sum_l \gamma_l)\gamma_l g_{t,l}(w)$ with $\mathcal{O}(LD)$ complexity, then calculate $\overline{g}_t$ with $\mathcal{O}(LD)$ complexity, which leads to $\mathcal{O}(LD)$ total complexity. **Second**, to update h_{t+1}, we calculate $\overline{h}_t$ with $\mathcal{O}(D)$ complexity, perform FFT on $\overline{h}_t$ and $\overline{g}_t$ with a complexity of $\mathcal{O}(D\log(D))$, calculate the multiplication of the FFT results with $\mathcal{O}(D)$ complexity, perform iFFT on the multiplication result with a complexity of $\mathcal{O}(D\log(D))$, and finally add $(h_t - \overline{h}_t)$ with $\mathcal{O}(D)$ complexity. In conclusion, updating h_{t+1} requires a complexity of $\mathcal{O}(D\log(D))$. **Finally**, in Line 23, we find θ_t by binary search with complexity $\mathcal{O}(\log(D))$ and update ϕ_t with complexity $\mathcal{O}(1)$ (since $h_t(w) = 0$ when $w < 0$). In summary, the complexity of Line 22 to update h_{t+1} for each task t is $\mathcal{O}(D\log(D))$, and the complexity of Line 23 to update θ_{t+1} and ϕ_{t+1} for each task is $\mathcal{O}(\log(D))$. As a result, the total complexity of the baseline-calibration phase for each task t is $\mathcal{O}(D\log(D))$.

In conclusion, the complexity of the OWA algorithm for each task during the online process is $\mathcal{O}(D\log(D))$.

E PROOF OF THEOREMS 2–6 AND COROLLARY 1

E.1 THEOREM 2

(Sufficient Resource) With γ_l^* from solving $\mathtt{P}_1$ in Eqs. (8)–(11), the OWA algorithm satisfies the resource sufficiency condition.

Proof. To prove Theorem 2, we require several lemmas. For convenience of presentation, we define $\gamma^* = \sum_l \gamma_l^*$. Since PDFs $g_{t,l}(\cdot)$ are continuous, we have $\phi_t = \gamma^*$ when $\theta_{t,m} > 0$. As a result, it holds that

$$h_{t+1}(w) = \overline{h}_t(w) * \overline{g}_t(w) + h_t(w) - \overline{h}_t(w). \tag{33}$$

Let us define $H_t(w) = \int_{-\infty}^{w} h_t(v)dv$, $G_{t,l}(w) = \int_{-\infty}^{w} g_{t,l}(v)dv$, and $\overline{G}_t(w) = \int_{-\infty}^{w} \overline{g}_t(v)dv$.

We first show that $H_{t+1}(\theta_t) \leq H_t(\theta_t)$.

Lemma 1. *$H_{t+1}(\theta_t) \leq H_t(\theta_t)$, $\forall t \in [T]$.*

Proof. If $\theta_t = 0$, according to the update rule of h_t and by the fact that $r_{t,l} > 0$, we have $H_{t+1}(0) = (1 - \gamma^* \phi_t) H_t(0) < H_t(0)$. If $\theta_t > 0$, by the definition of $H_t(w)$, we have

$$H_{t+1}(\theta_t) = \int_0^{\theta_t} h_{t+1}(w) dw \tag{34}$$

$$= \int_0^{\theta_t} \left(\overline{h}_t(w) * \overline{g}_t(w) + h_t(w) - \overline{h}_t(w) \right) dw. \tag{35}$$

Then by the definition of $\overline{h}_t(w)$, we have $h_t(w) - \overline{h}_t(w) = 0$ when $w \leq \theta_t$. Hence we have $H_{t+1}(\theta_t) = \int_0^{\theta_t} \overline{h}_t(w) * \overline{g}_t(w) dw$. Since $\int_0^{\theta_t} \overline{h}_t(w) dw = \int_0^{\theta_t} h_t(w) dw = H_t(\theta_t)$, by Fubini's theorem, we have

$$\int_{-\infty}^{\infty} \overline{h}_t(w) * \overline{g}_t(w) dw = \left(\int_{-\infty}^{\infty} \overline{h}_t(w) dw \right) \cdot \left(\int_{-\infty}^{\infty} \overline{g}_t(w) dw \right) \tag{36}$$

$$= H_t(\theta_t). \tag{37}$$

It is straightforward that $\overline{h}_t(w) \geq 0$ and $\overline{g}_t(w) \geq 0$, so we have

$$H_{t+1}(\theta_t) = \int_0^{\theta_t} \overline{h}_t(w) * \overline{g}_t(w) dw \leq H_t(\theta_t). \tag{38}$$

Thus, Lemma 1 is proved. □

Next, we show that $\theta_t \leq \theta_{t+1}$.

Lemma 2. *$\theta_t \leq \theta_{t+1}$ for every $t \in [T]$*

Proof. To prove Lemma 2, we first show three properties for H_t and θ_t:

- (i) CDF $H_t(w)$ is continuous and non-decreasing for $w > 0$. This is because $H_t(w)$ is the integral of $h_t(w)$, where $h_t(w)$ has a limited number of removable discontinuities at $w > 0$, and we have $h_1(w) \geq 0$ at every w.
- (ii) We have $\theta_t \geq 0$ for every t. This is because $h_t(w) = 0$ when $w < 0$.

Then we prove Lemma 2 by discussing separately the two cases where $\theta_t = 0$ and $\theta_t > 0$. When $\theta_t = 0$, by property (ii) we have $\theta_{t+1} \geq 0$, leading to $\theta_{t+1} \geq \theta_t$. When $\theta_t > 0$, by Lemma 1 we have $H_{t+1}(\theta_t) \leq H_t(\theta_t)$. Then by property (i), we have $\theta_{t+1} \geq \theta_t$. Thus, Lemma 2 is proved.

□

Next, we define $\overline{\Theta}_t = \min_w \{H_t(w) = 1\}$ and prove that $\overline{\Theta}_{t+1} \leq \theta_t + 1$.

Lemma 3. *$\overline{\Theta}_{t+1} \leq \theta_t + 1$ for every $t \in [T]$.*

Proof. We prove this lemma by induction. We start from the base case at $t = 1$. By the fact that $\theta_1 = 0$, $\phi_1 = 1$, the update rule of h_t and the convolution property of $\delta(w)$, we have $h_2(w) = (1 - \gamma^*)\delta(w) + \gamma^* \overline{g}_t(w)$. Since $\overline{G}_t(1) = 1$ for every t, we have $H_2(1) = 1$. As a result, $\overline{\Theta}_2 \leq 1 = \theta_1 + 1$. The base case is proved.

Suppose that $\overline{\Theta}_{t+1} \leq \theta_t + 1$ is valid at every $t \in [\tau - 1]$. We next prove that $\bar{\Theta}_{\tau+1} \leq \theta_\tau + 1$. We first show that there exists w such that $H_{\tau+1}(w) = 1$. By Fubini's theorem, when $\theta_\tau = 0$, we have

$$\int_{-\infty}^{\infty} h_{\tau+1}(w) dw = (1 - \gamma^*/\phi_\tau) \int_{-\infty}^{\infty} h_\tau(w) dw + (\gamma^*/\phi_\tau) \int_{-\infty}^{\infty} h_\tau(w) dw \tag{39}$$

$$= 1. \tag{40}$$

When $\theta_\tau > 0$, for $h_{\tau+1}$ we have

$$\begin{aligned}\int_{-\infty}^{\infty} h_{\tau+1}(w)dw &= \int_{-\infty}^{\infty} \left(\overline{h}_\tau(w) * \overline{g}_\tau(w) + h_\tau(w) - \overline{h}_\tau(w)\right) dw && (41)\\ &= \left(\int_{-\infty}^{\infty} \overline{h}_\tau(w)dw\right) \cdot \left(\int_{-\infty}^{\infty} \overline{g}_\tau(w)dw\right) + \int_{-\infty}^{\infty} h_\tau(w)dw - \int_{-\infty}^{\infty} \overline{h}_\tau(w)dw && (42)\\ &= \int_{-\infty}^{\infty} h_\tau(w)dw && (43)\end{aligned}$$

We define $\tau' = \max_{\tau'' \in [\tau+1]}\{\theta_{\tau''} = 0\}$, and by Lemma 2, we have $\tau' < \tau$. Then by the same derivation of Eqs. (41)–(43), we have

$$\begin{aligned}\int_{-\infty}^{\infty} h_{\tau+1}(w)dw &= \int_{-\infty}^{\infty} h_\tau(w)dw && (44)\\ &= \int_{-\infty}^{\infty} h_{\tau-1}(w)dw && (45)\\ &= \cdots && (46)\\ &= \int_{-\infty}^{\infty} h_{\tau'+1}(w)dw && (47)\\ &=1, && (48)\end{aligned}$$

where the last equation comes from Eq. (40).

Second, we show that when $w \geq \theta_\tau + 1$, we have $h_{\tau+1}(w) = 0$. When $\theta_\tau = 0$, by the update rule of h_t, we have $h_{\tau+1}(w) = (1-\gamma^*/\phi_\tau)h_\tau(w) + (\gamma^*/\phi_\tau)(\phi_\tau \cdot \delta(w) * \overline{g}_\tau(w) + h_\tau(w) - \overline{h}_\tau(w))$, since $\phi_\tau = \int_{-\infty}^{0} h_\tau(w)dw = \int_{-\infty}^{0} \overline{h}_\tau(w)dw$. Therefore, by the convolution property of $\delta(w)$, we have $h_{\tau+1}(w) = (1-\gamma^*/\phi_\tau)h_\tau(w) + (\gamma^*/\phi_\tau)(\phi_\tau \cdot \overline{g}_\tau(w) + h_\tau(w) - \overline{h}_\tau(w)$. Since $h_\tau(w)$ and $\overline{g}_\tau(w)$ become 0 when $w \geq 1$, so does $h_{\tau+1}(w)$. When $\theta_\tau > 0$, according to the definition of convolution and that fact that $\overline{h}_\tau(w) = \overline{g}_\tau(w) = 0$ at $w < 0$, we have

$$\overline{h}_\tau(w) * \overline{g}_\tau(w) = \int_0^w \overline{h}_\tau(v)\overline{g}_\tau(w-v)dv. \tag{49}$$

When $v \leq w-1$, we have $w - v \geq 1$ and $\overline{g}_\tau(w-v) = 0$, so

$$\overline{h}_\tau(w) * \overline{g}_\tau(w) = \int_{w-1}^w \overline{h}_\tau(v)\overline{g}_\tau(w-v)dv. \tag{50}$$

By the definition of $\overline{h}_\tau$, when $w \geq \theta_\tau + 1$, we have $\overline{h}_\tau(v) = 0$ at $v \in (w-1, w]$, so $\overline{h}_\tau(v) \cdot \overline{g}_\tau(w-v) = 0$. As a result, when $w \geq \theta_\tau + 1$, we have $h_{\tau+1}(w) = h_\tau(w)$ by the update rule of $h_{\tau+1}$.

Next, by Lemma 1, $H_\tau(\theta_{\tau-1}) \leq H_{\tau-1}(\theta_{\tau-1})$. We note that since $H_\tau(w)$ is the integral of $h_\tau(w)$, it continuously increases to 1 on $w > 0$. Then, ① since θ_τ is at least 0 for every τ, when $\theta_{\tau-1} = 0$, we have $\theta_\tau \geq \theta_{\tau-1}$; and ② when $\theta_{\tau-1} > 0$, we have $H_{\tau-1}(\theta_{\tau-1}) = \gamma^*$ and $\theta_\tau \geq \theta_{\tau-1}$. Then, since by the induction hypothesis, $H_\tau(\theta_{\tau-1}+1) = 1$, we have $h_\tau(w) = 0$ when $w \geq \theta_\tau + 1 \geq \theta_{\tau-1} + 1$. Therefore, we have $h_{\tau+1}(w) = 0$ when $w \geq \theta_\tau + 1$. Since ① there exists w that $H_{\tau+1}(w) = 1$, ② $H_{\tau+1}(w) = \int_{-\infty}^{w} h(v)dv$, ③ $\overline{\Theta}_t = \min_w\{H_t(w) = 1\}$, and ④ $h_{\tau+1}(w) = 0$ when $w \geq \theta_\tau + 1$, we have $\overline{\Theta}_{\tau+1} \leq \theta_\tau + 1$. Thus, Lemma 3 is proved. □

Now we show that $\theta_{t+1} < \theta_t + 1$.

Lemma 4. $\theta_{t+1} < \theta_t + 1$ *for every* $t \in [T]$.

Proof. Since ① by Lemma 3 we have $H_{t+1}(\theta_t + 1) = 1$, ② by the definition of θ_{t+1}, we have $H_{t+1}(\theta_{t+1}) \in [\gamma^*, 1]$, and ③ $H_{t+1}(w)$ is non-decreasing in w, we have $\theta_{t+1} \leq \theta_t + 1$. Furthermore, if $\theta_{t+1} = \theta_t = 0$, we have $\theta_{t+1} < \theta_t + 1 = 1$. And if $\theta_{t+1} > 0$, since $H_{t+1}(w)$ is continuous on $w > 0$, we have $H_{t+1}(\theta_{t+1}) = \gamma^* < 1$. As a result, we also have $\theta_{t+1} < \theta_t + 1$. Thus, Lemma 4 is proved. □

Next, we prove an inequality on H_t.

Lemma 5. *For every t we have*

$$H_t(w-1) \leq \gamma^* H_t(w), \forall w \in (-\infty, \overline{\Theta}_t). \tag{51}$$

Proof. First, if $\theta_{t-1} = 0$, then $\overline{\Theta}_t \leq 1$. For $w < \overline{\Theta}_t$, we have $w - 1 < 0$ and $H_t(w-1) = 0$. Therefore, Eq. (51) holds.

Next, we consider $\theta_{t-1} > 0$. We first derive the expression of H_t from h_t. According to the update rule of h_t, we have

$$\begin{aligned} H_{t+1}(w) &= \int_0^w \left(\int_0^v \overline{h}_t(u)\overline{g}_t(v-u)du + h_t(v) - \overline{h}_t(v) \right) dv \\ &= \int_0^w \int_0^v \overline{h}_t(u)\overline{g}_t(v-u)dudv + H_t(w) - \min\{H_t(w), H_t(\theta_t)\}. \end{aligned} \tag{52}$$

We define $\overline{H}_t(w) = \min\{H_t(w), H_t(\theta_t)\}$, and it is straightforward that $\overline{H}_t(w) = \int_0^w \overline{h}_t(v)dv$. Then from Eq. (53) and by the differential property of convolution, we have

$$H_{t+1}(w) = \int_0^w \overline{H}_t(w-v)\overline{g}_t(v)dv + H_t(w) - \overline{H}_t(w). \tag{53}$$

Now, we prove Eq. (51) by induction. Consider the first t' such that $\theta_{t'} > 0$. Since we have proved that Eq. (51) holds when $\theta_{t-1} = 0$, our base case is that Eq. (51) holds for all $t \leq t'$. Next, we assume that Eq. (51) holds for H_t at every $t \in [\tau]$, we prove that it also holds for $t = \tau + 1$.

When $w \leq \theta_\tau$, since $\overline{g}_\tau(w-v) = 0$ when $v \leq w-1$, and since $H_\tau(v) = \overline{H}_\tau(v)$ when $v \leq w$, we have

$$H_{\tau+1}(w) = \int_{w-1}^{w} H_\tau(v)\overline{g}_\tau(w-v)dv \tag{54}$$

and

$$H_{\tau+1}(w-1) = \int_{w-2}^{w-1} H_\tau(v)\overline{g}_\tau(w-1-v)dv \tag{55}$$

$$= \int_{w-1}^{w} H_\tau(v-1)\overline{g}_\tau(w-v)dv \tag{56}$$

$$\leq \gamma^* H_{\tau+1}(w), \tag{57}$$

where the last inequality comes from the induction hypothesis.

When $w \in (\theta_\tau, \overline{\Theta}_\tau)$, by Lemma 2 and Lemma 3 we have $\overline{\Theta}_\tau - 1 \leq \theta_\tau$. As a result,

$$H_{\tau+1}(w) = \int_{w-1}^{w} \overline{H}_\tau(v)\overline{g}_\tau(w-v)dv + H_\tau(w) - \overline{H}_\tau(w) \tag{58}$$

$$= \int_{w-1}^{\theta_\tau} H_\tau(v)\overline{g}_\tau(w-v)dv + \int_{\theta_\tau}^{w} H_\tau(\theta_\tau)\overline{g}_\tau(w-v)dv \tag{59}$$

$$+ H_\tau(w)\int_{w-1}^{w} \overline{g}_\tau(w-x)dx - \int_{w-1}^{w} H_\tau(\theta_\tau)\overline{g}_\tau(w-v)dv \tag{60}$$

$$= \int_{w-1}^{\theta_\tau} H_\tau(v)\overline{g}_\tau(w-v)dv + H_\tau(w)\int_{w-1}^{w} \overline{g}_\tau(w-x)dx - \int_{w-1}^{\theta_\tau} H_\tau(\theta_\tau)\overline{g}_\tau(w-v)dv \tag{61}$$

$$\geq \int_{w-1}^{w} H_\tau(v)\overline{g}_\tau(w-v)dv + \int_{w-1}^{\theta_\tau} (H_\tau(w) - H_\tau(\theta_\tau)) \cdot \overline{g}_\tau(w-v)dv \tag{62}$$

$$\geq \int_{w-1}^{w} H_\tau(v)\overline{g}_\tau(w-v)dv \tag{63}$$

and

$$H_{\tau+1}(w-1) = \int_{w-2}^{w-1} H_\tau(v)\overline{g}_\tau(w-1-v)dv$$

$$=\int_{w-1}^{w} H_\tau(v-1)\overline{g}_\tau(w-v)dv \tag{64}$$
$$\leq\gamma^*\int_{w-1}^{w} H_\tau(v)\overline{g}_\tau(w-v)dv \tag{65}$$
$$\leq\gamma^* H_{\tau+1}(w). \tag{66}$$

When $w \in [\Theta_\tau, \Theta_{\tau+1})$, we have $w-1 \leq \theta_\tau$. Therefore,

$$H_{\tau+1}(w) = \int_{w-1}^{w} \overline{H}_\tau(v)\overline{g}_\tau(w-v)dv + H_\tau(w) - \overline{H}_\tau(w) \tag{67}$$
$$= \int_{w-1}^{\theta_\tau} H_\tau(v)\overline{g}_\tau(w-v)dv + \int_{\theta_\tau}^{w} H_\tau(\theta_\tau)\overline{g}_\tau(w-v)dv + H_\tau(w) \tag{68}$$
$$- \int_{w-1}^{w} \overline{H}_\tau(\theta_\tau)\overline{g}_\tau(w-v)dv \tag{69}$$
$$\geq \int_{w-1}^{w} H_\tau(v)\overline{g}_\tau(w-v)dv \tag{70}$$
$$= \int_{w-1}^{\Theta_\tau} H_\tau(v)\overline{g}_\tau(w-v)dv + \int_{\Theta_\tau}^{w} H_\tau(v)\overline{g}_\tau(w-v)dv \tag{71}$$
$$= \int_{w-1}^{\Theta_\tau} H_\tau(v)\overline{g}_\tau(w-v)dv + \int_{\Theta_\tau}^{w} \overline{g}_\tau(w-v)dv. \tag{72}$$

Meanwhile, by the induction hypothesis,

$$H_{\tau+1}(w-1) = \int_{w-2}^{w-1} \overline{H}_\tau(v)\overline{g}_\tau(w-1-v)dv \tag{73}$$
$$\leq \int_{w-1}^{\Theta_\tau} \overline{H}_\tau(v-1)\overline{g}_\tau(w-v)dv + \int_{\Theta_\tau}^{w} \overline{H}_\tau(v-1)\overline{g}_\tau(w-v)dv. \tag{74}$$

Since $\Theta_\tau - 1 \leq \theta_{\tau-1} \leq \theta_\tau$, and $\Theta_{\tau+1} - 1 \leq \theta_\tau$,

$$H_{\tau+1}(w-1) \leq\gamma^* \int_{w-1}^{\Theta_\tau} H_\tau(v)\overline{g}_\tau(w-v)dv + \gamma^* \int_{\Theta_\tau}^{w} \overline{g}_\tau(w-v)dv \tag{75}$$
$$=\gamma^* H_{\tau+1}(w). \tag{76}$$

And Lemma 5 is proved. □

Now we start to prove Theorem 2, i.e., we prove that $K \geq (1-\sum_{l\in[L]}\gamma_l^*)^{-1}(1-\max_l \gamma_l^* L)^{-1}$ satisfies the resource sufficiency condition.

To meet the resource sufficiency condition, we only need to make sure that $\theta_t + 1 \leq K$ for every t. At $t = 1$, we have $\theta_1 = 0$ and $K \geq 1$, so $\theta_1 + 1 \leq K$ regardless of γ_l^*. For $t \in [2, T]$, by Lemma 3 we have $H_{t+1}(\theta_t + 1) = 1$. Then we can express $\theta_t + 1$ by

$$\theta_t + 1 = \int_0^{\theta_t+1} H_{t+1}(\theta_t+1)dw \tag{77}$$
$$= \int_0^{\theta_t+1} H_{t+1}(\theta_t+1) - H_{t+1}(w) + H_{t+1}(w)dw \tag{78}$$
$$= \int_0^{\theta_t+1} H_{t+1}(w)dw + \int_0^{\theta_t+1} H_{t+1}(\theta_t+1) - H_{t+1}(w)dw. \tag{79}$$

We define $d_t = \int_0^{\theta_t+1} H_{t+1}(w)dw$ and $d_t' = \int_0^{\theta_t+1} H_{t+1}(\theta_t+1) - H_{t+1}(w)dw$, then we have

$$\theta_t + 1 = d_t + d_t'. \tag{80}$$

Now we first seek an upper bound on d_t. By Lemma 2 and Lemma 4, we have

$$\begin{aligned}
d_t =& \int_0^{\theta_t+1} H_{t+1}(w)dw && (81)\\
=& \int_0^{\theta_t} H_{t+1}(w)dw + \int_{\theta_t}^{\theta_{t+1}} H_{t+1}(w)dw + \int_{\theta_{t+1}}^{\theta_t+1} H_{t+1}(w)dw && (82)\\
\le& \int_0^{\theta_t} H_{t+1}(w)dw + \gamma^*(\theta_{t+1}-\theta_t) + (\theta_t+1-\theta_{t+1}) && (83)\\
=& \int_0^{\theta_t} H_{t+1}(w)dw + \gamma^*(\theta_{t+1}-\theta_t) - (\theta_{t+1}-\theta_t) + 1 && (84)\\
\le& \int_0^{\theta_t} H_{t+1}(w)dw + 1. && (85)
\end{aligned}$$

Then, by Lemma 5 and the fact that $H_t(w)$ is continuous and non-decreasing on $w \ge 0$, we have

$$\begin{aligned}
d_t \le& \int_0^{\theta_t} H_{t+1}(w)dw + 1 && (86)\\
=& \sum_{k=0,1\dots\lfloor\theta_t\rfloor} \int_{\theta_t-k-1}^{\theta_t-k} H_{t+1}(w)dw + 1 && (87)\\
\le& \sum_{k=0,1\dots\lfloor\theta_t\rfloor} H_{t+1}(\theta_t-k) + 1 && (88)\\
\le& \sum_{k=0,1\dots\lfloor\theta_t\rfloor} (\gamma^*)^k \cdot \gamma^* + 1 && (89)\\
=& \frac{\gamma^*(1-(\gamma^*)^{\lfloor\theta_t\rfloor})}{1-\gamma^*} + 1 && (90)\\
\le& \frac{\gamma^*}{1-\gamma^*} + 1 && (91)\\
=& \frac{1}{1-\gamma^*}. && (92)
\end{aligned}$$

Meanwhile, for d'_t, we have

$$\begin{aligned}
d'_t =& \int_0^{\theta_t+1} H_{t+1}(\theta_t+1) - H_{t+1}(w)dw && (93)\\
=& \int_0^{\theta_t+1} \int_w^{\theta_t+1} h_{t+1}(x)dxdw && (94)\\
=& \int_0^{\theta_t+1} \int_0^{v} h_{t+1}(v)dwdv && (95)\\
=& \int_0^{\theta_t+1} vh_{t+1}(v)dv. && (96)
\end{aligned}$$

When $\theta_t = 0$, we can bound d'_t by

$$\begin{aligned}
d'_t =& \int_0^{\theta_t+1} vh_{t+1}(v)dv && (97)\\
=& \int_0^{1} vh_{t+1}(v)dv && (98)\\
=& \int_0^{1} v\left[(1-\frac{\gamma^*}{\phi_t})h_t(v) + \frac{\gamma^*}{\phi_t}\left(\int_0^v \overline{h}_t(w)\overline{g}_t(v-w)dw + h_t(v) - \overline{h}_t(v)\right)\right]dv && (99)
\end{aligned}$$

$$\leq \int_0^1 v \left[(1-\frac{\gamma^*}{\phi_t})h_t(v) + \frac{\gamma^*}{\phi_t}\left(\int_0^v \overline{h}_t(w)\overline{g}_t(v-w)dw + h_t(v)\right)\right] dv \tag{100}$$

$$= \int_0^1 v \left[\frac{\gamma^*}{\phi_t}\left(\int_0^v \overline{h}_t(w)\overline{g}_t(v-w)dw\right) + h_t(v)\right] dv \tag{101}$$

$$= \frac{\gamma^*}{\phi_t}\int_0^1 v \int_0^v \overline{h}_t(w)\overline{g}_t(v-w)dwdv + \int_0^1 vh_t(v)dv. \tag{102}$$

Since $\theta_t \leq \theta_{t+1}$ and $\phi_t = \int_{-\infty}^0 h_t(w)dw = \int_{-\infty}^0 \overline{h}_t(w)dw$, we have $\overline{h}_t(w) = \phi_t\delta(w)$ and we obtain the following recursive form to bound d'_t:

$$d'_t = \frac{\gamma^*}{\phi_t}\int_0^1 v \int_0^v \overline{h}_t(w)\overline{g}_t(v-w)dwdv + d'_{t-1} \tag{103}$$

$$= \frac{\gamma^*}{\phi_t}\int_0^1 v\phi_t\overline{g}_t(v)dv + d'_{t-1} \tag{104}$$

$$= \gamma^* \int_0^1 v\frac{\sum_l \gamma_l^* g_{t,l}(v)}{\gamma^*}dv + d'_{t-1} \tag{105}$$

$$\leq \sum_l (\gamma_l^*\mathbb{E}[r_{t,l}]) + d'_{t-1}. \tag{106}$$

When $\theta_t > 0$, noting that $\int_0^1 vh_1(w)dw = 0$, we have

$$d'_t = \int_0^{\theta_t+1} vh_{t+1}(v)dv \tag{107}$$

$$= \left(\int_0^{\theta_t} + \int_{\theta_t}^{\theta_t+1}\right) vh_{t+1}(v)dv \tag{108}$$

$$= \int_0^{\theta_t} v\left[\int_0^v h_t(w)\overline{g}_t(v-w)dw\right]dv + \int_{\theta_t}^{\theta_t+1} v\left[\int_0^{\theta_t} h_t(w)\overline{g}_t(v-w)dw + h_t(v)\right]dv \tag{109}$$

$$= \int_0^{\theta_t}\int_w^{\theta_t} vh_t(w)\overline{g}_t(v-w)dvdw + \int_0^{\theta_t}\int_{\theta_t}^{\theta_t+1} vh_t(w)\overline{g}_t(v-w)dvdw + \int_{\theta_t}^{\theta_t+1} vh_t(v)dv. \tag{110}$$

By setting $l = v - w$, we can rewrite the above as

$$d'_t = \int_0^{\theta_t}\int_0^{\theta_t+1-w} (l+w)h_t(w)\overline{g}_t(l)dldw + \int_{\theta_t}^{\theta_t+1} vh_t(v)dv. \tag{111}$$

Note that for $w \in [0, \theta_t]$, we have $\theta_t + 1 - w \geq 1$. Furthermore, $\overline{g}_t(l) = 0$ when $l \geq 1$. Therefore, we have

$$d'_t = \int_0^{\theta_t}\int_0^1 (l+w)h_t(w)\overline{g}_t(l)dldw + \int_{\theta_t}^{\theta_t+1} vh_t(v)dv \tag{112}$$

$$= \int_0^{\theta_t}\int_0^1 lh_t(w)\overline{g}_t(l)dldw + \int_0^{\theta_t}\int_0^1 wh_t(w)\overline{g}_t(l)dldw + \int_{\theta_t}^{\theta_t+1} vh_t(v)dv. \tag{113}$$

By Lemmas 2–4, $h_t(v) = 0$ when $v \geq \theta_{t-1} + 1$. Besides, since $\int_0^1 \overline{g}_t(l)dl = 1$, we have:

$$d'_t = \int_0^{\theta_t} h_t(w)\frac{\sum_l \gamma_l^*\mathbb{E}[r_{t,l}]}{\gamma^*}dw + \int_0^{\theta_t+1} vh_t(v)dv \tag{114}$$

$$= \sum_l \gamma_l^*\mathbb{E}[r_{t,l}] + d'_{t-1}. \tag{115}$$

By Eqs (106) and (115), we have $d'_t \leq \sum_l \gamma_l^* \sum_{t'\in[T]} \mathbb{E}[r_{t',l}] \leq \max_l \gamma_l^* KL$. By Eq. (92), we finally have

$$\theta_t + 1 = d'_t + d_t \leq \frac{1}{1-\gamma^*} + \max_l \gamma_l^* KL. \tag{116}$$

In optimization problem $\mathtt{P}_1$, constraint (9) is equivalent to $K \geq \frac{1}{1-\sum_l \gamma_l} + \max_l \gamma_l KL$, which leads to $K \geq \theta_t + 1$. As a result, the γ_l^* solved from $\mathtt{P}_1$ satisfies $K \geq \theta_t + 1$ when $\Theta_t \leq \theta_t$, which meets the resource sufficiency condition. Thus, Theorem 2 is proved. □

E.2 THEOREM 3

(Worker Assignment Probability Lower Bound) For all l, the probability of processing each task t by worker l is no less than γ_l^*.

Proof. We use $h_t'(w)$ to denote the PDF of Θ_t and use $H_t'(w)$ to denote its CDF. We prove Theorem 3 by proving that $h_t'(w) \geq h_t(w)$ by induction.

It is straightforward that $h_1'(w) = h_1(w)$, so $h_1'(w) \geq h_1(w)$ and $H_t'(w) \geq H_t(w)$ is valid for $t = 1$. We assume that $h_t'(w) \geq h_t(w)$ is valid for every $t \in [1, \tau]$, and we show that it is also valid for $t = \tau + 1$.

If $\theta_\tau = 0$, we have

$$\begin{aligned} h_{\tau+1}'(w) =& (1 - \sum_{l\in[L]} \gamma_l^*/\phi_\tau) h_\tau'(w) + (\sum_{l\in[L]} \gamma_l^*/\phi_\tau) \left[\int_0^w \overline{h}_\tau'(v)\overline{g}_\tau(w-v)dv + h_\tau'(w) - \overline{h}_\tau'(w)\right] && (117)\\ \geq& h_{\tau+1}(w), && (118)\end{aligned}$$

where

$$\overline{h}_\tau'(w) = \begin{cases} h_\tau'(w), \ w \leq \theta_\tau \\ 0, \ w > \theta_\tau. \end{cases}$$

If $\theta_\tau > 0$, we have

$$\begin{aligned} h_{\tau+1}'(w) =& \int_0^w \overline{h}_\tau'(v)\overline{g}_\tau(w-v)dv + h_\tau'(w) - \overline{h}_\tau'(w) && (119)\\ \geq& h_{\tau+1}(w). && (120)\end{aligned}$$

We note that the evolution of h_τ' in Eqs. (117) and (119) is not simply duplicated from the update rule of h_τ, but derived from the operation of OWA. The term $\sum_l \gamma_l^*/\phi_\tau$ in Eq. (117) is the probability that we accept a task when $\theta_\tau = 0$ and $\Theta_\tau \leq \theta_\tau$, and the convolution represents the sum of two random variables.

Now that we have proved that $h_t'(w) \geq h_t(w)$, we finally prove Theorem 3. When $\theta_t = 0$, since $H_t(0) = \phi_t$, the probability of processing task t by worker l is $\frac{\gamma_l^*}{\phi_t} H_t'(0) \geq \frac{\gamma_l^*}{\phi_t} H_t(0) = \gamma_l^*$. When $\theta_t > 0$, the probability of processing task t with worker l is $\frac{\gamma_l^*}{\gamma} H_t'(\theta_t) \geq \frac{\gamma_l^*}{\gamma} H_t(\theta_t) \geq \gamma_l^*$. Thus, Theorem 3 is proved. □

E.3 THEOREM 4

(Competitive Ratio) The competitive ratio achieved by the OWA algorithm is at least

$$\alpha = \mathtt{P}_1(\{\gamma_l^*\}). \tag{121}$$

Proof. By Theorem 3, the expected total reward of OWA is $\mathbb{E}[\text{ALG}] = \sum_t \sum_l \gamma_l^* u_{t,l}$. The total reward of the offline optimal algorithm is at most $\sum_t \sum_l x_{t,l}^* u_{t,l}$, where $x_{t,l}^*$ is the decision of the offline optimal solution. Therefore, the competitive ratio of OWA is

$$\begin{aligned} \frac{\mathbb{E}[\text{ALG}]}{\max \text{OPT}} \geq& \frac{\sum_t \sum_l \gamma_l^* u_{t,l}}{\sum_t \sum_l x_{t,l}^* u_{t,l}} && (122)\\ \geq& \min_t \frac{\sum_l \gamma_l^* u_{t,l}}{\sum_l x_{t,l}^* u_{t,l}} && (123)\\ \geq& \min_t \min_l \frac{\sum_{l'} \gamma_{l'}^* u_{t,l'}}{u_{t,l}} && (124)\end{aligned}$$

$$= \min_t \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma^*_{l'} u_{t,l'}}{u_{t,l}} + \gamma^*_l \tag{125}$$

$$\geq \min_t \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma^*_{l'} \underline{u}_{l'}}{\overline{u}_l} + \gamma^*_l \tag{126}$$

$$= \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma^*_{l'} \underline{u}_{l'}}{\overline{u}_l} + \gamma^*_l . \tag{127}$$

□

E.4 COROLLARY 1

(Closed-Form Lower Bound on the Competitive Ratio) The competitive ratio of OWA is lower bounded by $\alpha' = \max\{1/L, c\} \cdot (1 - K^{-\frac{1}{2}})$, where $c = \min_{l \in [L]} \underline{u}_l / \max_{l \in [L]} \overline{u}_l$ is a constant derived from the problem instance.

Proof. We find a set of feasible solutions for problem $\mathtt{P}_1$, and show that they achieve a competitive ratio of α' on GMPMW. It is straightforward that the constant solution $\gamma_l = \gamma' = \frac{1-K^{-\frac{1}{2}}}{L}$ for all l satisfies every constraint of $\mathtt{P}_1$. With this feasible solution, the competitive ratio satisfies

$$\frac{\mathbb{E}[\text{ALG}]}{\max \text{OPT}} \geq \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma' \underline{u}_{l'}}{\overline{u}_l} + \gamma' \tag{128}$$

$$\geq \gamma' \tag{129}$$

$$= \frac{1 - K^{-\frac{1}{2}}}{L} . \tag{130}$$

In addition, denote $\underline{u}_{\min} = \min_{l \in [L]} \underline{u}_l$ and $\overline{u}_{\max} = \max_{l \in [L]} \overline{u}_l$, then the competitive ratio also satisfies

$$\frac{\mathbb{E}[\text{ALG}]}{\max \text{OPT}} \geq \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma' \underline{u}_{l'}}{\overline{u}_l} + \gamma' \tag{131}$$

$$\geq \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma' \underline{u}_{\min}}{\overline{u}_l} + \gamma' \tag{132}$$

$$\geq \min_l \frac{\sum_{l' \in [L] \setminus \{l\}} \gamma' \underline{u}_{\min}}{\overline{u}_{\max}} + \gamma' \tag{133}$$

$$= \gamma' c (L-1) + \gamma' \tag{134}$$

$$\geq \gamma' c L \tag{135}$$

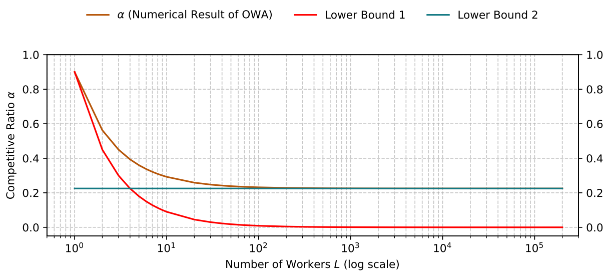
$$= c \cdot (1 - K^{-\frac{1}{2}}). \tag{136}$$

By Eqs. (130) and (136), we have $\frac{\mathbb{E}[\text{ALG}]}{\max \text{OPT}} \geq (1 - K^{-\frac{1}{2}}) \cdot \max\{1/L, c\} = \alpha'$. Thus, Corollary 1 is proved. □

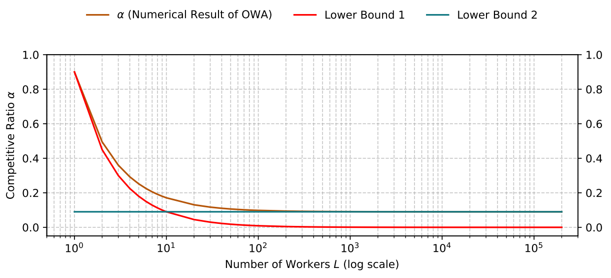
E.5 THEOREM 6

(Asymptotic Optimality of α') When the reward lower bound of each worker is 0, α' is an asymptotically optimal competitive ratio for the OWA algorithm on GMPMW, meaning that α' approaches the maximum possible competitive ratio when budget K approaches infinity.

Proof. We only need to show that there exists an adversary for which no online algorithm can achieve a competitive ratio higher than $\alpha^* = \frac{1}{L}$ when $c = 0$. We focus on the task sequence where the lower and upper bounds of the reward for each l are $\underline{u}_l = 0$ and $\overline{u}_l = u$. First, we have that the sum of the average probabilities that an online algorithm ALG processes a task by each worker l is no more than 1. If ALG assigns tasks to some worker l with an average probability greater than $1/L$, then there must exist some worker l' for which the probability that ALG assigns tasks to it is less than $1/L$. Then the adversary arranges the task sequence as follows: the reward for processing each task by worker l' is set to u, and the reward for processing each task by other workers $l \neq l'$ is set to 0. Then the offline optimal reward is exactly Tu, and the expected total reward of ALG is less than Tu/L.



(a) $c = 0.25$.



(b) $c = 0.1$.

Figure 5: Numerical result on α.

Next, we consider the case where the average probabilities of ALG processing a task by each worker are all $1/L$. First, if the probability that each worker l processes each task t is $\frac{1}{L}$ (i.e., $\Pr\{x_{t,l} = 1\} = \frac{1}{L}$), then the performance ratio of ALG is exactly $\frac{1}{L}$. Next, for an arbitrary worker l', if for any t', $\Pr\{x_{t',l'} = 1\} < \frac{1}{L}$, the adversary can arrange the reward by setting $u_{t',l'} = u$ and $u_{t,l} = 0$ for every other $t \neq t'$ and $l \neq l'$, so that the competitive ratio achieved by ALG is exactly $\Pr\{x_{t',l'} = 1\} < \frac{1}{L}$. On the other hand, if for some t', ALG has $\{x_{t',l'} = 1\} > \frac{1}{L}$, there must exist some other t'' where $\{x_{t'',l'} = 1\} < \frac{1}{L}$, since the average probabilities of ALG processing a task by this worker is $1/L$. In this case, the adversary arranges the reward by setting $u_{t'',l'} = u$ and $u_{t,l} = 0$ for every other $t \neq t''$ or $l \neq l'$, so that the performance ratio achieved by ALG is $\Pr\{x_{t'',l'} = 1\} < \frac{1}{L}$.

Now we have proved that $\alpha^* = \frac{1}{L}$ is the optimal competitive ratio that any online algorithm ALG can achieve on GMPMW. When $c = 0$, since $\lim_{K\to\infty} \alpha' = \frac{1}{L}$, α' is an asymptotically optimal competitive ratio. □

F NUMERICAL RESULTS ON COMPETITIVE RATIO α

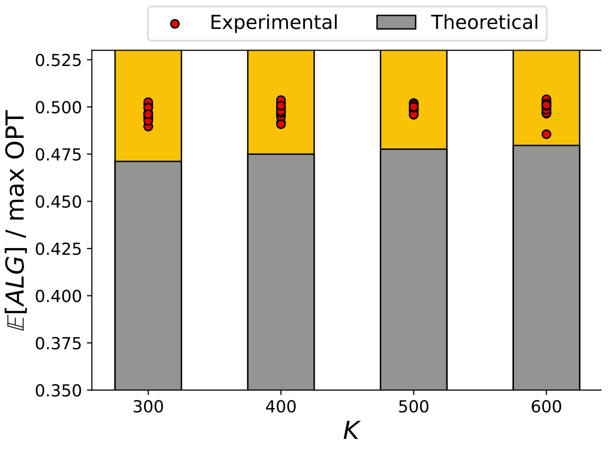
In this section, we present numerical results on the competitive ratio α in Figure 5 and compare them with the lower bound α'. Specifically, Figure 5 shows the numerical results of our competitive ratio $\mathtt{P}_1(\{\gamma_l^*\})$ (brown lines) alongside two components of the lower bound: the first part, $\frac{1}{L}(1 - K^{-\frac{1}{2}})$ (red lines, labeled as "Lower Bound 1"), and the second part, $c(1 - K^{-\frac{1}{2}})$ (green lines, labeled as "Lower Bound 2"), evaluated under varying values of c and different numbers of workers L.

In Figure 5, the reward upper bound $\overline{u}_l$ for each worker is set to 1. In Figure 5a, the reward lower bound $\underline{u}_l$ for each worker is 0.25, so the constant c equals 0.25. In Figure 5b, the reward lower bound $\underline{u}_l$ for each worker is 0.1, so there is $c = 0.1$. As shown in Figure 5, when L is small, the numerical competitive ratio α is well bounded by and stays close to $\frac{1}{L}(1 - K^{-\frac{1}{2}})$ (red line), which corresponds to the first part of the lower bound α'. As L increases, the numerical competitive ratio α approaches but remains above $c(1 - K^{-\frac{1}{2}})$ (green line), representing the second part of the lower bound α'. Consequently, Figure 5 illustrates that the competitive ratio α of the OWA algorithm consistently exceeds its lower bound α', while α' closely approximates α by capturing the dominant factors that influence its value for both small and large numbers of workers.

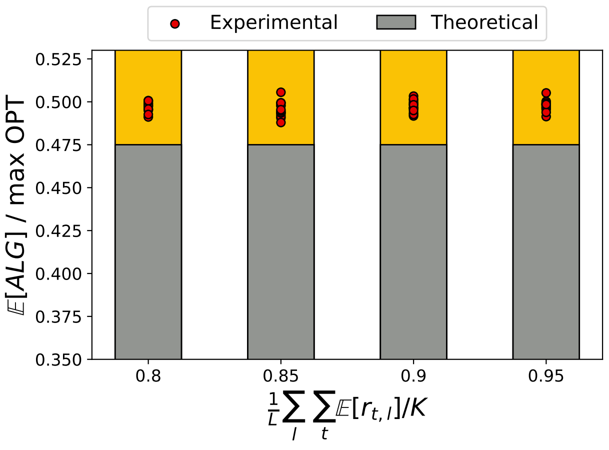
G EXPERIMENTAL EVALUATION OF THE COMPETITIVE RATIO

In Figure 6 we show the ratio between the performance of the OWA algorithm, i.e., $\mathbb{E}[\text{ALG}]$, and the optimal offline result $\max \text{OPT}$. Recall that $\max \text{OPT}$ is calculated as $\max \text{OPT} = \sum_t \max_l u_{t,l}$. In each simulation round, we generate one task sequence, and can directly obtain $\max \text{OPT}$. In each realization of the task sequence, $\overline{u}_l$ is independently set uniformly randomly in the interval from 0 to 100 for each $l \in [L]$ and $\underline{u}_l$ is set to 0. We generate 10 realizations of the task sequence to estimate $\mathbb{E}[\text{ALG}]$ of the OWA algorithm. We plot the ratio $\frac{\mathbb{E}[\text{ALG}]}{\max[\text{OPT}]}$ as one red point in Figure 6. We plot 15 points for each bar.

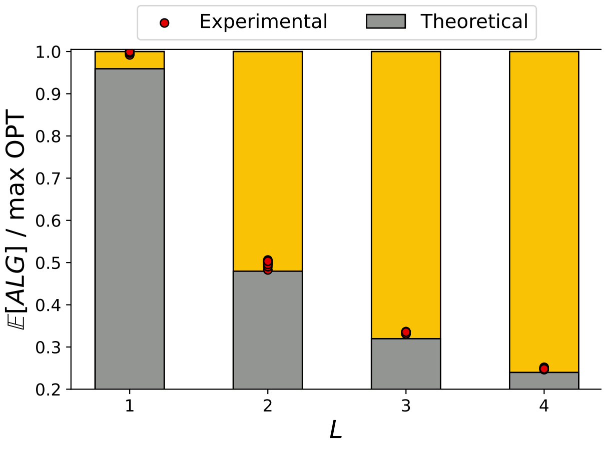
The expectation of the distribution $\mathcal{R}_{t,l}$ are set to $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}] = K$ for the base case, and $\mathcal{R}_{t,l}$ are generated as follows. For the task sequence of each simulation round, we randomly generate a demand factor a_l from $(0, 1]$ for each



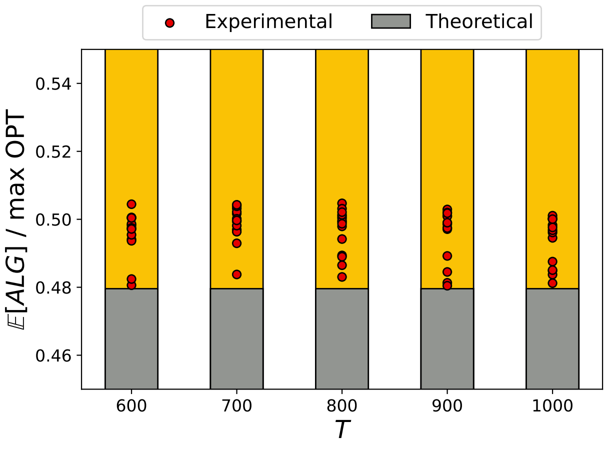
(a) Impact of K.



(b) Impact of $\frac{1}{L}\sum_l \sum_t \mathbb{E}\left[r_{t,l}\right]/K$.



(c) Impact of L.



(d) Impact of T.

Figure 6: Impact of various parameters on the performance of the OWA algorithm.

worker, then set the expected overall consumption of each worker to $K_l = \frac{a_l K L}{\sum_l a_l}$. The granularity of each PDF $g_{t,l}$ is set to 0.1. We generate $\mathbb{E}[r_{t,l}]$, for each t and l, uniformly on $(0,1)$. Then we normalize them so that $\sum_t \mathbb{E}[r_{t,l}] = K_l$. Then, if $\mathbb{E}[r_{t,l}]$ is greater than 0.5, we set $\Pr\{r_{t,l} = 1\} = (\mathbb{E}[r_{t,l}] - 0.5)/0.5$ and $\Pr\{r_{t,l} = 0.5\} = 1 - \Pr\{r_{t,l} = 1\}$; if $\mathbb{E}[r_{t,l}] \in [0.1, 0.5)$, we set $\Pr\{r_{t,l} = 0.5\} = (\mathbb{E}[r_{t,l}] - 0.1)/0.4$ and $\Pr\{r_{t,l} = 0.1\} = 1 - \Pr\{r_{t,l} = 0.5\}$; and if $\mathbb{E}[r_{t,l}] < 0.1$, we set $\Pr\{r_{t,l} = 0\} = 1$. This approach guarantees that the resource consumption follows our assumptions that $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}] \le K$ and $r_{t,l} \le 1$.

In Figure 6a, we investigate the competitive ratio of the OWA algorithm under different resource budgets K. We let the resource budget K increase from 300 to 600 while setting $T = 1000$ and $L = 2$. We observe that the theoretical competitive ratio and the actual performance increase as K increases.

In Figure 6b, we investigate the competitive ratio of the OWA algorithm under different $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}]/K$, which represents the ratio between the expected resource consumption and the resource budget. We set $K = 400$, $T = 1000$, and $L = 2$. The ratio $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}]/K$ is set to 0.8, 0.85, 0.9, and 0.95, with the randomly generated total consumption K_l of each worker scaled accordingly. In GMPMW, the ratio $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}]/K$ can be interpreted as the scarcity of the resource budget, and is no larger than 1 due to the assumption that $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}] \le K$. In this case, a larger ratio of $\frac{1}{L}\sum_l \sum_t \mathbb{E}[r_{t,l}]/K$ indicates that, on average, more resource is required to process all tasks. We observe that the theoretical competitive ratio and the actual performance of OWA are not affected by this ratio, demonstrating that the OWA algorithm is insensitive to fluctuations in resource demand, which is consistent with our theoretical result.

In Figure 6c, we investigate the competitive ratio of the OWA algorithm with different numbers of workers L. We increase L from 1 to 4, while fixing T=1000 and K=600. We observe that the theoretical competitive ratio and the actual performance of OWA decrease as L increases.

In Figure 6d, we investigate the competitive ratio of the OWA algorithm with different T values. In this set of experiments, we fix $K = 600$ and $L = 2$. We increase T from 500 to 1000. We observe that the theoretical competitive ratio and the actual performance of OWA are insensitive to the length of the task sequence T, which is consistent with the theoretical result.

In all cases of the above experiments, we further notice that the lowest red dots are usually close to the gray bars while remaining above them. This is consistent with the proven tightness of our theoretical bound.

H PERFORMANCE OF THE OWA ALGORITHM IN CASE STUDY

Besides deriving performance bounds for the OWA algorithm in terms of its competitive ratio, we also study its real-world performance against benchmarks. In this section, we evaluate the performance of OWA through a case study, with trace-driven experiments on real-time video analytics with multiple deployed machine learning models.

H.1 MODEL SELECTION FOR REAL-TIME VIDEO ANALYTICS

The rapid development of machine learning and edge computing has made it possible to deploy real-time video analytics on edge devices. Real-time video analysis requires sophisticated machine learning models, but the edge devices, such as tablets and laptops, are equipped with limited batteries. In many situations, the edge devices are not connected to a persistent power supply, so it is important to efficiently manage the workload on the edge devices.

Here we consider a general scenario where an edge computing device, equipped with multiple machine learning models, is used for real-time video analytics, processing a sequence of video chunks in practical applications such as traffic monitoring in smart cities and hazard prevention. Different machine learning models (workers) generate different accuracy values (reward) and consume different amounts of energy (resource). We need to decide whether to process or discard a video chunk and, if so, which machine learning model to use, in order to maximize the overall accuracy (equivalently, the average accuracy) within the energy constraint.

This application scenario of real-time video analysis with multiple models deployed on the edge device is consistent with the formulation of GMPMW. The video chunks are the incoming tasks, and the multiple machine learning models to process each video chunk are the multiple workers in GMPMW. The probability distribution of the energy consumption of each model $\mathcal{R}_{t,l}$ can be estimated by model profiling and video pre-processing [Hung et al., 2018], and the upper bound $\overline{u}_l$ and lower bound $\underline{u}_l$ can also be obtained by profiling [Zhang et al., 2017]. The edge device does not know the accuracy and energy consumption of processing each video chunk before that chunk arrives, and only after processing a chunk using a model can it know the corresponding accuracy and energy consumption. The energy budget of the edge device is the limited resource budget.

Video Traces For the video traces, we use a Xiaomi 12 Pro Android smartphone equipped with a Sony IMX766 photosensor to capture the video content as well as the network traces. The smartphone is mounted on a moving vehicle and positioned to capture a comprehensive view of the traffic on the road. We collect 4 sets of video traces, labeled as Trace 1, Trace 2, Trace 3, and Trace 4. The length of Trace 1 is 1200 seconds, the length of Trace 2 is 1800 seconds, the length of Trace 3 is 2400 seconds, and the length of Trace 4 is 3000 seconds. The video frames are grouped into video chunks, with each video chunk containing the video frames of 3 seconds. The video chunks are then sent to the edge device for analysis, so each video chunk is a task.

Model Profiling (Energy Consumption) As illustrated in Figure 7, we deploy 3 machine learning models on a laptop computer powered by an Intel Core i5-11320H CPU with integrated graphics, to analyze the video chunks. The machine learning models deployed are Faster R-CNN [Ren et al., 2017], YOLOv5 [Jocher, 2020] with a medium backbone, and YOLOv5 with a large backbone. To obtain the maximum and minimum accuracy and energy consumption of each model, we deploy them on the same laptop as above.

We profile the energy consumption distribution for the models (edge devices) on a 300-second video. The video is captured by the same Xiaomi 12 Pro smartphone and streamed to the laptops via the HTTP Live Streaming protocol using the FFmpeg software, simulated using the network traces. The resolution of the video varies among {360p, 540p, 720p} as a result of the bitrate adaptation under different network conditions. To evaluate power consumption, we use the Python implementation of Intel's Running Average Power Limit (pyRAPL) [Spirals, 2023]. The pyRAPL library connects Intel's hardware-level power counters, facilitating the measurement of power consumed by CPU cores and DRAM. This integration enables accurate, real-time monitoring of energy consumption throughout our experiments. By regularly sampling energy data, pyRAPL provides valuable insights into the energy efficiency of various computational tasks, allowing us to establish correlations between specific software actions and corresponding energy consumption.

Model Profiling (Accuracy) We use the accuracy of processing each video chunk by each model as the reward. To effectively benchmark and compare the performance of different object detection models, it is critical to define a consistent metric for accuracy. Each inference generated by an object detection model yields a set of predicted bounding boxes. In parallel, we construct a set of ground truth objects, each represented by its own bounding box. These ground truth bounding boxes are derived from a highly reliable model, the Faster R-CNN [Ren et al., 2017], which uses the Resnet 50 [He et al., 2016] backbone.

The accuracy of our model is determined by comparing these two sets of bounding boxes, as illustrated in Figure 7. This evaluation is quantified by the Intersection over Union (IoU) metric. For each object, IoU represents the ratio of the intersection area to the union area. Specifically, the intersection area is defined as the overlap between the predicted bounding box and the ground truth bounding box, while the union area encompasses both bounding boxes in their entirety. Thus, we have

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{137}$$

for each object. This metric is a standard and widely accepted method in video analytics [Lin et al., 2014, Redmon et al., 2016, Ren et al., 2017], as it provides a clear measure of the accuracy of the bounding boxes proposed by the model.

Let IoU_i represent the IoU of the i-th bounding box. The total number of detected objects is denoted by Det, while GT (Ground-Truth) represents the number of ground-truth objects. The accuracy of the model is then quantified as the ratio of the sum of all IoUs to the total number of ground-truth objects. This can be expressed as

$$\text{Accuracy} = \frac{\sum_{i=1}^{\text{Det}} \text{IoU}_i}{\text{GT}}. \tag{138}$$

The upper and lower bounds of the accuracy of each model are profiled in the same environment as the energy consumption distributions for each model. The accuracy and energy measurements are executed once per second and averaged over each 3-second time slot. Note that even if a task is discarded, a small amount of accuracy can still be achieved. This is due to the temporal similarity between consecutive video chunks. By utilizing the inference results from the previous chunk, we can still gain some accuracy. For example, if we choose to discard chunk $t+1$, reapplying the results from chunk t can still contribute some accuracy. However, we do not allow the continuous reuse of these results. Should chunk $t+2$ also be discarded, the resulting accuracy would be set to zero.

Trace Parameter The results of our model profiling are as follows. The minimum accuracy of each model is 0. The maximum accuracy of Faster R-CNN is 0.9581, the maximum accuracy of YOLOv5 with a medium backbone is 0.9716, and the maximum accuracy of YOLOv5 with a large backbone is 0.9773. The average energy consumption to process one video chunk for Faster R-CNN, YOLOv5 with medium backbone, and YOLOv5 with large backbone are 6.4 J, 11.0 J, and 18.9 J, respectively. Accordingly, the average energy consumption of these models to process videos of 1200 s, 1800 s, 2400 s, and 3000 s is estimated as 4 kJ, 6 kJ, 8 kJ, and 10 kJ, respectively.

H.2 BENCHMARKS

We consider a wide range of benchmarks to compare with the OWA algorithm in the trace-driven experiments.

1. Random (R): The system decides to process each video chunk using each model l with a probability of $1/(L+1)$, and to discard it also with a probability of $1/(L+1)$.
2. Adaptive (Ada): If the ratio between the total number of tasks processed and the number of tasks that have arrived is less than 1 (since the resource budget is on average enough to process every task), the system randomly chooses a model l with a probability of $1/L$ to process the incoming task.

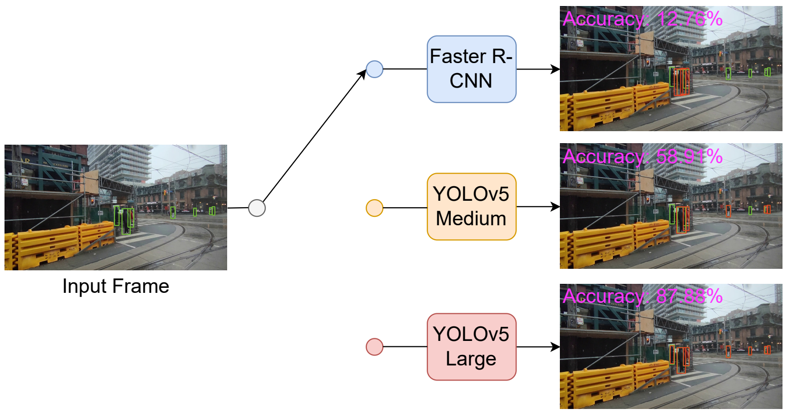
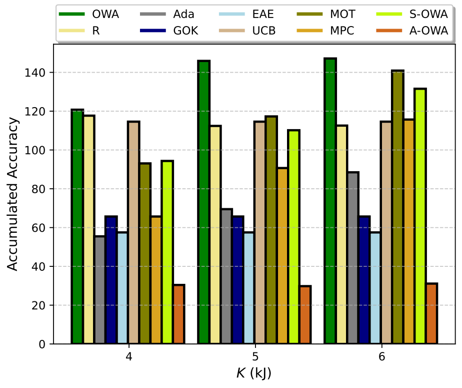
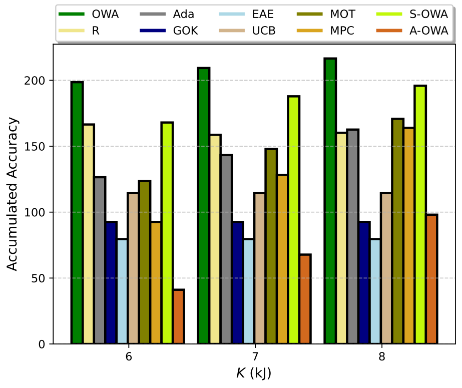


Figure 7: (Reproduced from Figure 1) Illustration of video analytics and IoU calculation. Green boxes are ground-truth results, and red boxes are profiling results.

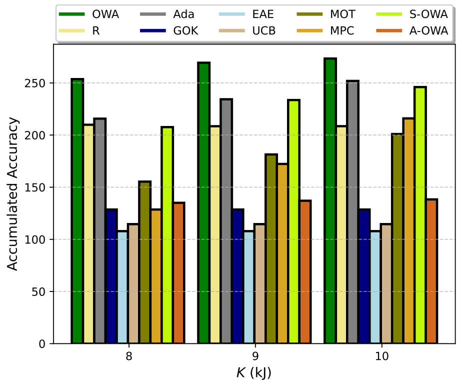
3. Greedy Online Knapsack (GOK): For each incoming task, the system calculates the average reward of each model as $(\underline{u}_l + \overline{u}_l)/2$, and calculates the average consumption of each model as $\mathbb{E}[r_{t,l}]$. The system then processes the incoming task that has the highest efficiency, which is calculated as the average reward divided by the average energy consumption.
4. Exploration and Exploitation (EAE) [Audibert et al., 2009]: The system repeatedly performs exploration and exploitation as follows: The system starts from the exploration at $t = 1$. In the exploration stage, the system uses each model l to process 5 tasks and discards 5 tasks, and memorizes each reward and resource consumption. Each time a task is processed by a model or discarded, the average reward and the average resource consumption of that model (or the discarding of a task) are updated. The average reward (resp. average resource consumption) of a model is calculated as the cumulative reward (resp. total resource consumption) of that model divided by the number of tasks processed by that model. The average reward of discarding a task is calculated as the cumulative reward of discarding a task divided by the number of tasks discarded. The average resource consumption of discarding a task is always 0. In total, the system processes $5(L+1)$ tasks in each exploration stage. After each exploration stage, the system enters the exploitation stage, which also lasts for $5(L+1)$ tasks. In the exploitation stage, for each incoming task, the system selects the model (or discards a task) with the highest efficiency, which is calculated by dividing the average reward by the average resource consumption. The system also updates the average reward and the average resource consumption of each model (or discarding a task) in the exploitation stage.
5. Upper Confidence Bound Bandit (UCB) [Garivier and Moulines, 2011]: For the first L chunks, the system processes the l-th chunk using model l. Then the system discards the $(L+1)$-st chunk. The system records and updates $Q_{t,l}$ as the average reward per task of model l, and uses $Q_{t,L+1}$ to record and update the average reward of discarding a task. The system records and updates $N_{t,l}$ as the number of tasks processed by model l, and uses $N_{t,L+1}$ to record and update the number of tasks discarded. Each time a new task t arrives, the system processes it with the model with the highest upper confidence bound, or discards it if the upper confidence bound for discarding a task is the highest. The upper confidence bound UCB_l is calculated as $\text{UCB}_l = Q_{t,l} + 1.75\left(\ln(t)/N_{t,l}\right)^{1/2}$.
6. Multi-Worker One-Way Trading (MOT): The MOT algorithm is a multi-worker variant of the algorithm designed for the One-Way Trading Problem (OTP) [Cao et al., 2020]. The system estimates the length of the task sequence as T' and operates under two situations: The first situation is $K - \Theta_t < T' - t$. In this case, when task t arrives, the system calculates the following threshold for each model l as $\int_{\Theta_t/K}^{\Theta_t/K + r_l^{avg}/K} \phi(x)dx$, where $\phi(x) = \frac{\overline{u}_l K}{2} + (\rho - \frac{\overline{u}_l K}{2})e^{\frac{x\rho\overline{u}_l K}{2}}$, r_l^{avg} is the average resource consumption of model l, $\rho = \frac{\overline{u}_l K}{2}(W(\frac{1}{e}) + 1)$, and W is the Lambert-W function. The system compares each $\overline{u}_l$ with this threshold, and chooses the worker with the largest $\overline{u}_l$ that exceeds its threshold to process task t. The second situation is $K - \Theta_t \geq T' - t$. In this case, the system will process the incoming task with the model that has the highest $\overline{u}_l$.
7. Model Predictive Control (MPC) [Morari and Lee, 1999]: The system predicts the reward and resource consumption of processing task t by each model using the history data of tasks $t-30$ to $t-1$. For each model l, if the system has processed at least one of the last 30 tasks with it, the efficiency of model l for task t is predicted as $(\sum_{\tau\in[t-10,t-1]} x_{\tau,l}u_{\tau,l})/\sum_{\tau\in[t-10,t-1]} x_{\tau,l}r_{\tau,l}$. Otherwise, the efficiency of model l for task t is predicted as



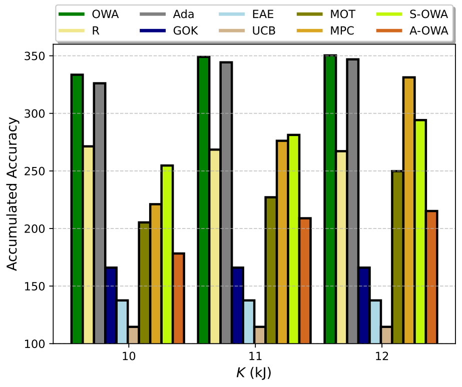
(a) Trace 1 (1200s).



(b) Trace 2 (1800s).



(c) Trace 3 (2400s).



(d) Trace 4 (3000s).

Figure 8: (Reproduced from Figure 2) Comparing the OWA algorithm against benchmarks on different traces.

$\frac{1}{2}(\underline{u}_l + \overline{u}_l)/\mathbb{E}[\mathcal{R}_{t,l}]$. The system will then process task t using the processing level with the highest predicted efficiency. If the current task $t < 30$ or the system has not processed any of the past 30 tasks, it randomly selects a model to process the incoming task.

8. Single-Worker OWA (S-OWA): The system implements a naive single-worker version of the OWA algorithm, focusing on the model with the largest $\overline{u}_l$. We denote this model by l^*. The system initializes $h_1(w) = \delta(w)$, updates $h_{t+1}(w)$ by $h_{t+1}(w) = (1-\gamma'/\phi_t)h_t(w)+(\gamma'/\phi_t)[\overline{h}_t(w)*g_{t,l^*}(w)+h_t(w)-\overline{h}_t(w)]$ if $\theta_t = 0$, and update by $h_{t+1}(w) = \overline{h}_t(w)*g_{t,l^*}(w) + h_t(w) - \overline{h}_t(w)$ otherwise, where the target θ_{t+1} is updated by $\theta_{t+1} = \arg\min_w\{\int_{-\infty}^{w} h_{t+1}(v)dv \geq \gamma'\}$ and γ' is set to $\gamma' = 1 - K^{-1/2}$. The system processes the task using this model if the resource consumption level Θ_t falls below the target θ_t, and discards it otherwise.

9. Average OWA (A-OWA): This is another naive variant of the OWA algorithm. The system keeps a different resource utilization function $h_{t,l}(w)$ and baseline $\theta_{t,l}$ for each model l. A-OWA initializes $h_{t,l}$ as $\delta(\cdot)$ and updates it by $h_{t+1,l}(w) = (1 - \sum_{l'} \gamma^*_{l'}/\phi_{t,l})h_{t,l}(w) + (\sum_{l'} \gamma^*_{l'}/\phi_{t,l})[\overline{h}_{t,l}(w) * g_{t,l}(w) + h_{t,l}(w) - \overline{h}_{t,l}(w)]$ if $\theta_{t,l} = 0$, and $h_{t+1,l}(w) = \overline{h}_{t,l}(w) * g_{t,l}(w) + h_{t,l}(w) - \overline{h}_{t,l}(w)$ otherwise, where $\overline{h}_{t,l}(w) = h_{t,l}(w)$ if $w \leq \theta_{t,l}$ and $\overline{h}_{t,l}(w) = 0$ otherwise. Furthermore, $\theta_{t,l}$ is initialized to 0 and updated as $\theta_{t,l} = \arg\min_w\{\int_{-\infty}^{w} h_{t,l}(v)dv \geq \sum_{l'} \gamma^*_{l'}\}$, and $\phi_{t,l}$ is initialized to 1 and updated by $\phi_{t,l} = \int_{-\infty}^{0} h_{t,l}(w)dw$. When each task arrives, the decision maker chooses model l with probability $\gamma^*_l / \sum_{l'} \gamma^*_{l'}$. If the remaining resource budget of the decision maker falls below $\theta_{t,l}$, model l will process that task. Otherwise, the incoming task will be discarded.

The accuracy of the R, Ada, MPC, S-OWA, and A-OWA algorithms is averaged over 10 runs, as they are randomized, while a single run is performed for the deterministic algorithms GOK, EAE, UCB, and MOT.

H.3 PERFORMANCE

To compare the performance of the OWA algorithm against that of the benchmarks, we apply all algorithms to each video trace with different resource budgets K, as shown in Figure 8.

In all settings, we observe that the OWA algorithm outperforms all benchmarks under all conditions on our video traces. We also have some observations on the performance of each of the benchmarks. We will discuss them in three groups: ① Random, Adaptive, and GOK; ② EAE, UCB, MOT, and MPC; and ③ S-OWA and A-OWA.

The Random algorithm performs worse than OWA because it does not consider the resource constraints. The adaptive algorithm considers the resource constraint, but simply controls the number of tasks (chunks) processed, rather than the resource consumption, so it performs worse than OWA. The GOK algorithm sticks to the worker (model) with the highest average efficiency, but the most efficient worker may not fully utilize all of the resource (energy), leading to inferior performance.

EAE and UCB perform worse than the OWA algorithm because they are not aware of the resource constraint. MOT performs worse than the OWA algorithm because, in our system, the reward (accuracy) and the resource consumption are not known when a video chunk arrives. However, these are important variables for the decision-making process in the MOT algorithm. When MOT can only use the profiled data to make decisions, its performance suffers. MPC performs worse than the OWA algorithm because the real-world street scenes captured in our video traces are highly fluctuating. As a result, the predictions made by MPC are not accurate.

Finally, we discuss S-OWA and A-OWA, which are different variants of OWA. S-OWA focuses only on the best possible worker (model) according to model profiling. It performs worse than OWA because it does not balance between the reward and resource consumption by utilizing all workers. On the other hand, A-OWA first chooses the worker (model) and then makes a decision based on that worker's baseline. This strategy of A-OWA leads to worse performance than OWA because each worker only updates its own baseline, but the different reward and resource consumption levels among different workers are coupled in GMPMW. In contrast, OWA updates the baselines in a joint manner and assigns each task to different workers based on the model profiling result, thus achieving superior performance.

In conclusion, our experimental results demonstrate the excellent capability of the OWA algorithm to utilize the multiple workers in GMPMW in a variety of realistic system settings, and they show the importance of properly handling the multiple workers in the proposed approach.