A Multi-Armed Bandit Approach to Online Resource Allocation for Service Platforms

Zihao Wang[1] and Wei You*[1]

[1]Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Hong Kong

Abstract

We study finite-horizon online resource allocation under uncertain matching rewards and hard server capacities. A linear-programming benchmark characterizes the full-information optimum, while tracking and probability sampling translate fractional allocation targets into pathwise-feasible assignments. For homogeneous customers, UCB with cumulative tracking achieves logarithmic instance-dependent regret and $O(\sqrt{JN \log N})$ worst-case regret, equivalently $\widetilde{O}(\sqrt{JN})$, and Thompson sampling with probability sampling achieves $O(\sqrt{N})$ Bayesian regret under realized-assignment feedback. We extend the model to a fixed panel of M latent-type customers, each contributing one task in each of N assignment rounds. Customer-level posterior learning yields a single-batch Bayesian regret bound of order $\sqrt{N} + (J-1)\sqrt{JN/M}$. For repeated batches, a clustering–modeling–assignment architecture estimates latent population structure from historical service data and converts the estimates into informative priors. An accelerated expectation-maximization procedure provides two-round high-probability recovery under identifiability and sufficient exploration. Numerical experiments show stable feasible execution under cumulative tracking and probability sampling, competitive UCB and Thompson-sampling performance in the homogeneous model, and substantial gains from informative latent-type priors in heterogeneous systems.

Keywords: resource allocation, regret minimization, multi-armed bandit, constrained bandit

1 Introduction

Digital service platforms increasingly operate as real-time allocation systems. Cloud platforms assign computational jobs to servers, online labor platforms route tasks to workers, and similar marketplaces match heterogeneous service requests to limited-capacity resources. In these environments, the platform typically knows its operational constraints but does not know in advance the reward from serving a particular customer on a particular server. Service outcomes therefore play a dual role: they generate immediate payoff and reveal information that should improve later assignments.

*Corresponding author: weiyou@ust.hk.

This creates a finite-horizon exploration–exploitation problem under hard capacity constraints. Aggressive exploitation may commit scarce resources before the platform has learned enough about customer–server compatibility, whereas excessive exploration can waste valuable capacity on low-value matches. The challenge is especially acute for modern platforms that serve many small users with uncertain and heterogeneous needs, rather than a small number of repeat customers with stable demand. A useful online policy must therefore learn and allocate simultaneously while remaining feasible at every step.

We study this problem through a linear-programming benchmark and two progressively richer models. The LP characterizes the full-information optimum when mean rewards are known and serves as the reference point for regret analysis. At the same time, implementing an LP target online is nontrivial: every task must be assigned to an available server, so feasibility must hold pathwise rather than only in expectation. This execution issue is central to our approach.

We analyze a homogeneous benchmark and a latent-type heterogeneous extension. In both settings, assignments are sequential, capacity constraints are hard, and rewards must be learned from service outcomes.

1.1 Main contributions

This paper makes three contributions.

First, we study a finite-horizon online resource-allocation problem for service platforms with homogeneous tasks, hard per-server capacities, and unknown matching rewards. We formulate an LP benchmark based on the full-information optimum and show that its target allocation can be implemented in a pathwise-feasible manner by an oracle tracking rule with only constant loss. Building on this execution layer, we develop two learning policies: UCB with cumulative tracking (UCB-CT) and Thompson sampling with probability sampling (TS-PS). UCB-CT achieves logarithmic instance-dependent regret together with an $O(\sqrt{JN \log N})$ worst-case regret bound, equivalently $\widetilde{O}(\sqrt{JN})$, while TS-PS achieves $O(\sqrt{N})$ Bayesian regret under realized-assignment feedback. The homogeneous model is a structured special case of stochastic BwK with deterministic identity consumption; our contribution here is not the formulation itself, but the feasibility-aware execution layer and the sharper regret analysis enabled by this structure.

Second, we extend the model to heterogeneous customers with latent types, taking the problem beyond the standard BwK. A batch consists of N assignment rounds; in every round, each of the M customers contributes one task. Because the same customer contributes repeatedly, observations from early assignments should inform later assignments for that customer. We capture this feature through a customer-level benchmark and posterior model, prove that the customer-level LP benchmark converges to the type-level LP benchmark as the batch size grows, and propose TS-PS-Heterogeneous. Relative to the customer-level benchmark, TS-PS-Heterogeneous attains a single-batch Bayesian regret bound of order $\sqrt{N} + (J-1)\sqrt{JN/M}$ under realized-assignment feedback. To support repeated deployment across batches, we further propose a clustering–modeling–assignment pipeline that estimates the latent reward matrix and type proportions from historical service data, converts

these estimates into informative priors, and reuses them in later batches. Under identifiability and sufficient exploration, our accelerated EM procedure recovers the realized clustering in two rounds with high probability and yields consistent estimates of the latent population parameters.

Third, we provide an extensive numerical study under a unified online resource-allocation template that separates reward estimation, LP optimization, and execution. This design lets us isolate why the algorithms work. The experiments show that cumulative tracking and probability sampling stabilize execution under scarce capacity and avoid the premature depletion of high-value servers that can arise under direct tracking. They also show that preserving latent structure matters: informative latent-type priors substantially improve performance and quickly approach the oracle benchmark, whereas pooling the same learned data into a non-latent warm start can perform worse than a cold start. Across the tested instances, Thompson-sampling-based policies are typically the most robust, especially when combined with informative priors.

1.2 Related work

Multi-armed bandit. The multi-armed bandit (MAB) problem (Lai et al., 1985; Gittins et al., 2011; Bubeck et al., 2012) is a foundational model for studying the exploration–exploitation trade-off. An agent repeatedly pulls arms and receives stochastic rewards, with the goal of maximizing cumulative reward. Because the reward distributions are unknown, the agent must balance exploration and exploitation. In our setting, the servers play the role of arms. Classical MAB admits two canonical algorithms: upper confidence bound (UCB) (Auer, 2002) and Thompson sampling (TS) (Agrawal and Goyal, 2012). Several generalizations are closely related to our setting. Anantharam et al. (1987) and Komiyama et al. (2015) study multi-armed bandits with multiple plays (MP-MAB), in which the agent selects a set of L optimal arms and knows L in advance. In our homogeneous setting, the customer effectively needs to learn a set of optimal servers, but the size of that set is unknown, which makes our problem harder than MP-MAB. The heterogeneous setting is harder still. Agrawal et al. (1989) study a model in which the joint reward vector of the arms is unknown but takes values in a known finite set, inducing dependence across arms and requiring the learner to identify an optimal stationary allocation. A key difference from their model is the presence of capacity constraints. Our results can also be extended to other bounded, correlated reward models. Pacchiano et al. (2021), Wu et al. (2015), and Saxena et al. (2020) study constrained variants of MAB, but their constraints are not imposed on the number of pulls of each arm, which is the key feature in our resource-allocation setting. Several papers (Cesa-Bianchi and Lugosi, 2012; Combes et al., 2015; Komiyama et al., 2015) study combinatorial bandits, but those algorithms become computationally prohibitive when the number of matching types is large in online service platforms.

Bandits with Knapsacks. Bandits with knapsacks (BwK) provides the most direct benchmark for our homogeneous model. In stochastic BwK, each action generates both a reward and a resource-consumption vector, and the learner maximizes cumulative reward subject to multiple budgets over a finite horizon (Badanidiyuru et al., 2018; Li et al., 2021). A key distinction from classical

multi-armed bandits is that the relevant benchmark is typically an LP or dynamic-policy benchmark rather than the single best arm, because a mixture of actions may strictly outperform any fixed arm under resource constraints (Badanidiyuru et al., 2018; Sankararaman and Slivkins, 2021). Badanidiyuru et al. (2018) introduced the BwK model and established near-optimal worst-case regret guarantees up to logarithmic factors via balanced-exploration and primal-dual methods. Agrawal and Devanur (2015) extended this line to linear contextual settings with global resource constraints. Agrawal et al. (2016) subsequently developed an oracle-efficient contextual algorithm for general policy classes and concave objectives. From a Bayesian perspective, Ferreira et al. (2018) combined Thompson sampling with an LP subroutine for online network revenue management and showed that the same recipe extends to general multi-armed bandits with resource constraints.

Problem-dependent guarantees in BwK are subtler than in unconstrained stochastic bandits. Flajolet and Jaillet (2015) obtained logarithmic regret bounds in several important settings, but their guarantees rely on additional structure and instance parameters. Li et al. (2021) later gave the first general problem-dependent logarithmic regret bound for BwK through a primal-dual analysis under a nondegeneracy assumption on the underlying LP. Sankararaman and Slivkins (2021) complemented this line by providing a sharp beyond-worst-case characterization for UcbBwK relative to the best fixed-distribution benchmark, showing that logarithmic regret is attainable only under substantial structure, whereas broader classes of instances admit lower bounds of order $\Omega(\sqrt{T})$.

Our homogeneous model is best viewed as a structured special case of stochastic BwK with deterministic identity consumption and a fixed horizon, so the main novelty is not the formulation itself. Rather, the contribution in the homogeneous setting lies in the feasibility-aware execution layer under hard per-server capacities and in the sharper problem-specific regret analysis enabled by this special structure. Our heterogeneous extension moves further away from standard BwK by combining customer-level heterogeneity, latent types, and batch-level data reuse, which are not captured by the classical homogeneous-resource formulation.

Dynamic pricing and online resource allocation. Our paper is also related to dynamic pricing and network revenue management with limited inventory; see Den Boer (2015) for a review. In that literature, the decision maker learns demand information while controlling sales under finite resources. In particular, Besbes and Zeevi (2009, 2012) study learning-and-control policies for dynamic pricing and blind network revenue management, and Ferreira et al. (2018) combine Thompson sampling with an LP subroutine for online network revenue management, explicitly connecting that approach to general resource-constrained bandit problems. More recent work also studies learning-and-control in network revenue management, including Chen and Shi (2023). Accordingly, our contribution is not that capacity-constrained learning is absent from prior work. Rather, our contribution is to study a service-allocation model with hard per-server quotas and pathwise feasibility, which motivates the tracking-based implementation in our algorithms. We further extend it to a heterogeneous latent-type setting that is not captured by the standard homogeneous pricing/BwK models.

Learning and control in service systems. A separate line of work integrates online learning with queueing, congestion, or scheduling considerations in service systems (Massoulié and Xu, 2016; Sun et al., 2023; Kim and Vojnovic, 2021). These papers study the interaction between learning and operational control when customers may wait, abandon, or experience congestion. By contrast, we abstract away queueing effects and focus on centralized finite-horizon assignment under hard service capacities. This abstraction allows us to isolate the learning-allocation trade-off and to develop algorithms that are directly implementable for small and medium-sized service platforms.

A useful benchmark for our heterogeneous model is the latent-structure line in matching and online learning. In Johari et al. (2021), the platform knows the finite latent type space, the population proportions, and the reward matrix, but must infer each arriving worker's realized latent type online under capacity constraints; in that regime, logarithmic instance-dependent regret is achievable. Our setting is substantially harder because the latent structure (the reward matrix and the type proportions) must be estimated from data and then reused across service batches. On the queueing/control side, Hsu et al. (2022) study online learning with adaptive control when clients are unlabeled and payoff vectors are uncertain, and prove a finite-horizon payoff-gap bound with a $\sqrt{\log N/N}$ learning term, while explicitly leaving open whether a logarithmic learning term is achievable in an instance-dependent model. More broadly, the latent-bandit and online-clustering literature shows that shared hidden structure can materially change achievable regret; see, e.g., Maillard and Mannor (2014); Gentile et al. (2014, 2017); Pal et al. (2023). A closely related operations-management example is dynamic pricing with online clustering, where cross-product pooling sharpens learning when each product has limited local data (Miao et al., 2022). Finally, Kalvit and Zeevi (2022) establish logarithmic regret in a different large-market matching model with unknown population structure and effectively unlimited supply, which suggests that sharp rates may remain possible once the relevant shared structure is learned.

1.3 Organization

The rest of the paper is organized as follows. We begin with the homogeneous-task setting. Section 2 introduces the model and defines regret. Section 3 presents the online resource-allocation algorithms and analyzes their regret bounds. Section 4 studies heterogeneous customers, extends the Thompson sampling algorithm, and introduces the three-stage architecture. Section 5 reports the numerical results for our algorithms and the myopic benchmark.

2 Homogeneous Model and Regret Benchmark

A service platform operates $J \geq 1$ heterogeneous servers and must process $N \geq 1$ homogeneous tasks, where $[J] = \{1, \ldots, J\}$ and $[N] = \{1, \ldots, N\}$. Server j has a known integer capacity $c_j \geq 1$, and the aggregate capacity satisfies $\sum_{j=1}^{J} c_j > N$. We write $\mu_j = c_j/N$ for the normalized capacity share. For asymptotic statements, $\boldsymbol{\mu}$ is fixed and N ranges over compatible horizons for which $N\mu_j \in \mathbb{N}$ for every j. The case $J = 1$ has zero regret, so the learning results concern $J \geq 2$. The

platform assigns tasks centrally and sequentially.

For each $j \in [J]$ and $n \in [N]$, let $X_j^n \in \{0,1\}$ denote the reward that would be obtained if server j were selected at round n. We assume that the random variables $\{X_j^n\}_{j\in[J],n\in[N]}$ are mutually independent and that $X_j^n \sim \text{Bernoulli}(r_j)$.[1] The mean-reward vector $\boldsymbol{r} = (r_1, \ldots, r_J)$ is unknown to the platform; ties are permitted unless a gap-dependent result explicitly assumes strict ordering.

At round n, the platform selects one available server j_n and then observes $X_{j_n}^n$. Let B_j^n denote the remaining capacity of server j at the beginning of round n. The initial capacity is $B_j^1 = c_j$. The capacity dynamics satisfy $B_j^{n+1} = B_j^n - \mathbb{1}(j_n = j)$, for $j \in [J]$ and $n \in [N]$. For each $j \in [J]$ and $n \in \{0\} \cup [N]$, define $N_j^n = \sum_{s=1}^n \mathbb{1}(j_s = j)$, with $N_j^0 = 0$. Then, for every $j \in [J]$ and $n \geq 1$, we have $c_j - B_j^n = N_j^{n-1}$.

Throughout the paper, finite argmax operations use the smallest-index maximizer, LPs use the lexicographically smallest optimizer in the displayed variable order, and feasibility repairs follow the same convention. Let $\mathcal{F}_0$ be trivial and $\mathcal{F}_n = \sigma(j_1, X_{j_1}^1, \ldots, j_n, X_{j_n}^n)$, for $n \in [N]$. Randomized policies may also use mutually independent seeds $U_1, \ldots, U_N$, independent of $\boldsymbol{r}$ and the potential rewards. An algorithm is *admissible* if j_n is $\mathcal{F}_{n-1} \vee \sigma(U_n)$-measurable and $B_{j_n}^n > 0$ almost surely for every n. A server is *available* at round n when its remaining capacity is positive.

The objective is to maximize the expected total reward $\mathbb{E}\left[\sum_{n=1}^N X_{j_n}^n\right]$. By linearity of expectation and the definition of N_j^N, this objective can be written as $\mathbb{E}\left[\sum_{n=1}^N X_{j_n}^n\right] = \sum_{j=1}^J r_j \mathbb{E}[N_j^N]$.

2.1 Full-Information Benchmark and Regret

If the platform knew $\boldsymbol{r}$ in advance, the optimal full-information benchmark would reduce to a static linear program over assignment fractions. Let p_j denote the fraction of tasks assigned to server j over the horizon. The constraint $p_j \leq \mu_j$ captures the capacity limit of server j. Let $\boldsymbol{p}^*$ be an optimal solution to

$$\begin{aligned} R^* = \max_{\boldsymbol{p}\in\mathbb{R}^J} \quad & N \sum_{j=1}^J p_j r_j \\ \text{s.t.} \quad & 0 \leq p_j \leq \mu_j, \quad j \in [J], \quad \sum_{j=1}^J p_j = 1. \end{aligned} \tag{1}$$

Proposition 1. *For any admissible algorithm, the expected total reward is at most R^*.*

Proof. Consider any admissible algorithm and define $p_j = \mathbb{E}[N_j^N]/N$ for each $j \in [J]$. Admissibility implies $N_j^N \leq c_j = N\mu_j$ almost surely for every $j \in [J]$, and $\sum_{j=1}^J N_j^N = N$ almost surely. Hence the vector $\boldsymbol{p}$ is feasible for (1). Its objective value is exactly the expected total reward of the algorithm, namely $\sum_{j=1}^J r_j \mathbb{E}[N_j^N]$. Therefore, the expected total reward of any admissible algorithm cannot exceed R^*. □

We therefore define the regret of an admissible algorithm under a fixed reward vector $\boldsymbol{r}$ as $R^* - \mathbb{E}\left[\sum_{n=1}^N X_{j_n}^n \mid \boldsymbol{r}\right]$. By Proposition 1, this quantity is always nonnegative. In the homogeneous

[1] We adopt Bernoulli rewards for ease of exposition. The same algorithmic ideas and regret analysis extend to bounded reward distributions with standard modifications.

setting, our goal is to approach the classical stochastic-bandit benchmarks as closely as the hard capacity constraints allow. In particular, our UCB-based analysis establishes logarithmic problem-dependent regret, while Section 3.3 studies Bayesian regret for Thompson sampling.

For the ordered LP characterization and the gap-dependent regret analysis, relabel the servers so that $r_1 > r_2 > \cdots > r_J$ and define

$$S_k = \sum_{i=1}^{k} \mu_i, \qquad k = 0, 1, \ldots, J, \qquad S_0 = 0, \quad \text{and} \quad \widetilde{j} = \max\{j \in \{0, \ldots, J\} : S_j \le 1\}.$$

Because $\sum_{j=1}^{J} \mu_j > 1$, we have $\widetilde{j} \le J - 1$. The unique full-information LP solution is therefore

$$p_j^* = \begin{cases} \mu_j, & j \le \widetilde{j}, \\ 1 - S_{\widetilde{j}}, & j = \widetilde{j} + 1, \\ 0, & j \ge \widetilde{j} + 2. \end{cases} \tag{2}$$

Lemma 1. *The unique optimal solution to* (1) *is given by* (2).

The homogeneous model is a structured special case of stochastic bandits with knapsacks (BwK). Indeed, if server j is viewed as an arm, then pulling arm j yields Bernoulli reward with mean r_j and deterministically consumes one unit of resource j and zero units of all other resources. Hence the LP benchmark coincides with the static relaxation of this special-case BwK instance.

The key distinction lies in the online execution layer: the platform must assign every arriving task to an available server, so admissibility must hold pathwise at every round. This motivates a feasibility-aware implementation, which we realize through a tracking-based execution layer. Beyond feasibility, our analysis is sharper than generic worst-case BwK guarantees because it exploits the explicit LP structure of this identity-consumption subclass. Specifically, Lemma 1 and the closed-form solution in (2) make this structure explicit. This additional structure yields stronger instance-dependent guarantees than one would obtain from a generic BwK analysis.

3 Algorithms for the Homogeneous Model

This section develops admissible algorithms for the homogeneous model. We begin with an oracle benchmark that knows the mean-reward vector $\boldsymbol{r}$ and incurs only constant regret relative to the LP benchmark. We then turn to the learning problem, in which $\boldsymbol{r}$ is unknown. In the unconstrained stochastic bandit setting, UCB and Thompson sampling achieve the standard optimal instance-dependent and Bayesian regret rates. We show that both ideas can be adapted to the present capacity-constrained setting while preserving the same benchmark rates in their respective senses.

3.1 Oracle Tracking Benchmark

Suppose that the platform knows the true mean-reward vector $\boldsymbol{r}$. It can then solve (1) and obtain an optimal allocation vector $\boldsymbol{p}^*$.

A natural way to implement $\boldsymbol{p}^*$ is *probability sampling* (PS), which selects server j with probability p_j^* in every round. This rule is feasible only in expectation. Indeed, if some server j is binding in the LP, meaning $p_j^* = \mu_j \in (0,1)$, then under PS $N_j^N \sim \text{Binomial}(N, p_j^*)$. Hence the overflow $(N_j^N - c_j)_+$ has expectation of order $\sqrt{N}$ by the central limit theorem. If overflow tasks are reassigned to lower-reward servers after server j is exhausted, the resulting regret can therefore be of order $\sqrt{N}$ whenever the marginal reward gap is positive. Thus, PS cannot achieve the logarithmic instance-dependent regret that we seek, although it remains compatible with the optimal $\sqrt{N}$ Bayesian benchmark studied in Section 3.3.

To obtain an admissible algorithm with smaller regret, we adopt the tracking idea from pure exploration (Garivier and Kaufmann, 2016; Degenne et al., 2019). Tracking has also proved useful for attaining optimal regret in structured bandits (Degenne et al., 2020). In the present oracle setting, the target allocation $\boldsymbol{p}^*$ to be tracked is time-invariant. At round n, the cumulative tracking algorithm selects a server for which the gap between the target *cumulative allocation* np_j^* and the realized allocation N_j^{n-1} is largest.[2] We call the resulting rule *oracle tracking*.

Algorithm 1 Oracle tracking

1: Solve (1) and obtain $\boldsymbol{p}^*$.
2: Initialize $N_j^0 = 0$ for all $j \in [J]$.
3: **for** $n = 1, \ldots, N$ **do**
4: Select $j_n \in \text{argmax}_{j\in[J]} \left\{ np_j^* - N_j^{n-1} \right\}$. ▷ Tracking
5: Assign one task to server j_n and observe $X_{j_n}^n$.
6: **end for**

The analysis uses the cumulative-tracking discrepancy bound of Degenne et al. (2020), stated as Lemma 9 in Appendix B.

Lemma 2. *Algorithm 1 is admissible. In particular,* $B_{j_n}^n > 0$ *for all* $n \in [N]$.

The following proposition shows that oracle tracking loses at most a constant amount relative to the LP benchmark.

Proposition 2. *Under Algorithm 1,*

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq \left(\sum_{i=2}^{J} \frac{1}{i}\right)\left(\sum_{j=1}^{J} r_j\right).$$

In particular, oracle tracking has $O(1)$ *regret.*

Oracle tracking shows that the LP solution can be implemented sequentially without violating feasibility and with only constant regret. This tracking mechanism will serve as the execution layer for the learning algorithms developed next.

[2] In cumulative tracking, the algorithm selects a server whose realized allocation deviates most from the target cumulative allocation $\sum_{s=1}^{n} p_j^s$. If the argmax is not a singleton, we use the standing smallest-index tie-breaking rule. In the oracle setting, the target allocation is constant over time, so the cumulative target allocation reduces to np_j^*. This coincides with *direct tracking*, which tracks the instantaneous target in the current period.

3.2 UCB with cumulative tracking

UCB with cumulative tracking (UCB-CT) combines horizon-based confidence indices with a cumulative-tracking execution rule. The construction makes admissibility transparent and supports both a gap-dependent logarithmic analysis and a gap-free worst-case analysis. The direct-tracking variant UCB-DT is analyzed in Appendix A.1.

UCB-CT assumes $N > J$ and begins with one pull from each server; when $N = J$, only this initialization phase is defined. For the remaining $T = N - J$ rounds, let $b_j = c_j - 1$ and $\bar{\mu}_j = b_j/T$ for $j \in [J]$. Since $\sum_{j=1}^{J} c_j > N$, we have $\sum_{j=1}^{J} b_j = \sum_{j=1}^{J} c_j - J > N - J = T$, so the residual LP is feasible. At each round, UCB-CT computes UCB indices, solves the LP, and then applies cumulative tracking to the resulting sequence of target vectors.

Algorithm 2 UCB with cumulative tracking (UCB-CT)

1: Assign one task to each server and collect reward $X_{j_n}^n$, i.e., $j_n = n$ for $n \in [J]$.
2: Set $T = N - J$, $b_j = c_j - 1$, and $\bar{\mu}_j = b_j/T$ for all $j \in [J]$.
3: **for** $n = J+1, \ldots, N$ **do**
4: Compute

$$C_j^n = \frac{\sum_{s=1}^{n-1} X_{j_s}^s \mathbb{1}(j_s = j)}{N_j^{n-1}} + \sqrt{\frac{2\log N}{N_j^{n-1}}}, \qquad j \in [J].$$

5: Compute $\boldsymbol{q}^n \in \operatorname{argmax}_{\boldsymbol{q}\in\mathbb{R}^J} \left\{ \sum_{j=1}^{J} q_j C_j^n : 0 \le q_j \le \bar{\mu}_j \text{ for all } j, \sum_{j=1}^{J} q_j = 1 \right\}$.
6: Assign a task to server $j_n \in \operatorname{argmax}_{j\in[J]} \left\{ \sum_{s=J+1}^{n} q_j^s - (N_j^{n-1} - 1) \right\}$ ▷ Cumulative tracking
7: Collect reward $X_{j_n}^n$.
8: **end for**

Lemma 3. *Under UCB-CT, it holds that $B_{j_n}^n > 0$ for all $n \in [N]$. In particular, UCB-CT is admissible.*

For the problem-dependent result only, after initial unit sampling, define the rank parameters

$$\bar{S}_k = \sum_{i=1}^{k} \bar{\mu}_i, \qquad \bar{S}_0 = 0, \qquad \widehat{j} = \max\{k \in \{0, \ldots, J\} : \bar{S}_k \le 1\}, \qquad \bar{\delta} = 1 - \bar{S}_{\widehat{j}}.$$

Thus $\widehat{j}$ is the number of servers that are saturated in the LP, and when $\bar{\delta} > 0$, the server $\widehat{j} + 1$ is the unique critical server. Let $h_J = \sum_{i=2}^{J} \frac{1}{i}$.

Theorem 1 (Gap-dependent regret for UCB-CT)**.** *Assume $r_1 > r_2 > \cdots > r_J$.*
(i) If $\widehat{j} = 0$, then

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \le J + Jh_J + \sum_{j=2}^{J} (r_1 - r_j) A_j(N) + \frac{2J}{N^2},$$

where

$$A_j(N) = h_J + \left\lceil \frac{8\log N}{(r_1 - r_j)^2} \right\rceil, \qquad j = 2, \ldots, J.$$

(ii) If $\widehat{j} \geq 1$ *and* $\bar{\delta} := 1 - \bar{S}_{\widehat{j}} \in (0, \bar{\mu}_{\widehat{j}+1})$*, let* $c = \widehat{j} + 1$ *and* $Z = \{c+1, \ldots, J\}$*. Then*

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq J + Jh_J + (r_1 - r_c)\left(A_c(N) + \sum_{j \in Z} A_j(N)\right) + \sum_{j \in Z}(r_c - r_j)A_j(N) + \frac{2J}{N^2},$$

where

$$A_c(N) = h_J + \left\lceil \frac{8 \log N}{(r_{\widehat{j}} - r_c)^2} \right\rceil, \qquad A_j(N) = h_J + \left\lceil \frac{8 \log N}{(r_c - r_j)^2} \right\rceil, \quad j \in Z.$$

(iii) If $\widehat{j} \geq 1$ *and* $\bar{\delta} = 0$*, let* $Z = \{\widehat{j} + 1, \ldots, J\}$*. Then*

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq J + Jh_J + (r_1 - r_{\widehat{j}}) \sum_{j \in Z} A_j(N) + \sum_{j \in Z}(r_{\widehat{j}} - r_j)A_j(N) + \frac{2J}{N^2},$$

where

$$A_j(N) = h_J + \left\lceil \frac{8 \log N}{(r_{\widehat{j}} - r_j)^2} \right\rceil, \quad j \in Z.$$

In particular, UCB-CT achieves problem-dependent regret of order $O(\log N)$*.*

Theorem 2 (Worst-case regret for UCB-CT)**.** *For any* $\boldsymbol{r} \in [0,1]^J$*, with ties allowed, under UCB-CT it holds that*

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq J + Jh_J + 4(h_J + 2)\sqrt{2JN \log N} + \frac{2J}{N^2}.$$

In particular, for fixed J*, UCB-CT achieves a worst-case regret bound of order* $O(\sqrt{JN \log N})$*; equivalently, the rate is* $\widetilde{O}(\sqrt{JN})$ *when logarithmic factors are suppressed.*

The proofs of Theorems 1 and 2 are both given in Appendix C. They are based on the same decomposition. After the warm start, the regret splits into a target-selection term and a tracking term. Cumulative tracking controls the second term uniformly by Jh_J. The first term is then handled either by rank arguments, which yield the logarithmic bound, or by a weighted optimism bound, which yields the gap-free worst-case guarantee.

Remark 1. *Appendix A studies the direct-tracking variant UCB-DT. That variant is the more direct feasibility-aware analogue of classical direct tracking and also admits an instance-dependent logarithmic bound. UCB-CT is presented in the main text because the cumulative-tracking discrepancy bound yields a common proof architecture for Theorems 1 and 2. A comparable gap-free theorem for UCB-DT would require additional control of the temporal variation of the empirical LP optimizer.*

3.3 A Thompson Sampling Algorithm

For the Bayesian formulation, suppose the mean-reward vector $\boldsymbol{r} = (r_1, \ldots, r_J)$ is drawn from a known prior distribution $\mathbb{Q}^1$. At each round n, the platform updates its posterior distribution $\mathbb{Q}^n = \mathbb{Q}(\cdot \mid \mathcal{F}_{n-1})$, where $\mathcal{F}_{n-1}$ is the history filtration defined in Section 2. Thompson sampling then balances exploration and exploitation by sampling a reward vector from the current posterior and optimizing as if that sampled vector were the truth.

Example 1 (Bernoulli rewards and Beta priors)**.** *Assume that* $X_j^n \sim \text{Bernoulli}(r_j)$ *for all* $j \in [J]$ *and* $n \in [N]$*. Suppose also that the prior factorizes as* $\mathbb{Q}^1 = \bigotimes_{j=1}^J Q_j^1$ *and* $Q_j^1 = \text{Beta}(\alpha_j, \beta_j)$*. Define* $W_j^{n-1} = \sum_{s=1}^{n-1} X_{j_s}^s \mathbb{1}(j_s = j)$ *and* $N_j^{n-1} = \sum_{s=1}^{n-1} \mathbb{1}(j_s = j)$*. Then the posterior also factorizes as* $\mathbb{Q}^n = \bigotimes_{j=1}^J Q_j^n$*, where* $Q_j^n = \text{Beta}(\alpha_j + W_j^{n-1}, \beta_j + N_j^{n-1} - W_j^{n-1})$*, for* $j \in [J]$*.*

At each round n, the platform samples a reward vector $\boldsymbol{S}^n$ from the posterior $\mathbb{Q}^n$. It then solves the LP benchmark with $\boldsymbol{S}^n$ in place of $\boldsymbol{r}$, obtaining an allocation vector $\boldsymbol{p}^n$. Next, it queries a server $\widehat{j}_n$ according to $\boldsymbol{p}^n$. If the queried server is available, the platform assigns the task to that server. Otherwise, it applies the tracking rule to $\boldsymbol{p}^n$ and selects an available server. The query is an internal randomization device: no outcome is observed from an unavailable queried server, and the posterior is updated only with the reward $X_{j_n}^n$ from the server that actually processes the task. We refer to this information structure as *realized-assignment feedback* and call the resulting policy Thompson sampling with probability sampling (TS-PS).

Algorithm 3 Thompson sampling with probability sampling (TS-PS)

Initialize $N_j^0 = 0$ and $B_j^1 = c_j$ for all $j \in [J]$.
for $n = 1, \ldots, N$ **do**
  Sample $\boldsymbol{S}^n \sim \mathbb{Q}^n$.
  Compute $\boldsymbol{p}^n \in \text{argmax}_{\boldsymbol{p} \in \mathbb{R}^J} \left\{ \sum_{j=1}^J p_j S_j^n : 0 \le p_j \le \mu_j \text{ for all } j, \sum_{j=1}^J p_j = 1 \right\}$.
  Query a server $\widehat{j}_n \sim \boldsymbol{p}^n$. ▷ Probability sampling
  if $B_{\widehat{j}_n}^n > 0$ **then**
    Set $j_n = \widehat{j}_n$.
  else
    Set $j_n \in \text{argmax}_{j \in [J]} \left\{ n p_j^n - N_j^{n-1} \right\}$.
  end if
  Observe $X_{j_n}^n$ and update the posterior $\mathbb{Q}^n \mapsto \mathbb{Q}^{n+1}$ using $(j_n, X_{j_n}^n)$.
  Update $N_j^n = N_j^{n-1} + \mathbb{1}(j_n = j)$ and $B_j^{n+1} = B_j^n - \mathbb{1}(j_n = j)$ for all $j \in [J]$.
end for

In Ferreira et al. (2018), the authors propose TS-Fixed for price-based network revenue management with limited inventory. TS-Fixed incorporates inventory constraints through an LP with fixed average inventory rates and then samples an action according to the resulting optimal distribution. However, unlike our setting, it does not impose hard action-level capacity constraints on how many times a given action can be selected. Therefore, TS-Fixed cannot be applied directly to our online resource allocation problem with per-server capacities.

Our TS-PS algorithm follows the same posterior-sampling-plus-LP template as TS-Fixed, but adds a tracking-based fallback step whenever the sampled server is unavailable. As shown in Section 3.1, simply sampling from the LP solution can violate the hard capacity constraints and thus lead to an inadmissible policy. The tracking step restores admissibility while steering the realized allocation toward the target proportions.

Lemma 4. *Under TS-PS, we have* $B_{j_n}^n > 0$ *for all* $n \in [N]$*; hence TS-PS is admissible.*

Because Thompson sampling is defined in a Bayesian setting, we evaluate its performance by Bayesian regret. For a fixed mean-reward vector $\boldsymbol{r}$, let

$$\mathrm{Regret}(N, \boldsymbol{r}) = R^*(\boldsymbol{r}) - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n \middle| \boldsymbol{r}\right].$$

The Bayesian regret is then defined by

$$\mathrm{BayesRegret}(N) = \mathbb{E}_{\boldsymbol{r}\sim\mathbb{Q}^1}\left[\mathrm{Regret}(N, \boldsymbol{r})\right].$$

Theorem 3. *Under TS-PS with realized-assignment feedback, we have*

$$\mathrm{BayesRegret}(N) \le \left(6\sqrt{J} + 12\sqrt{J\log J} + 3(J-1)\sqrt{J}\right)\sqrt{N}.$$

In particular, for fixed J, TS-PS achieves Bayesian regret of order $O(\sqrt{N})$.

The proof of Theorem 3 follows the high-level posterior-sampling strategy of Ferreira et al. (2018) and separates regret into a learning term and a feasibility-repair term. A repair-flow argument controls the number of assignments for which the sampled and realized servers differ. Proofs are given in Appendix D.

4 Systems with Heterogeneous Customers

Many service platforms must match heterogeneous customers to servers with limited capacity. Following Johari et al. (2021), we incorporate two salient features. First, each server has limited service capacity within a finite operating window. Second, customers are heterogeneous, so the reward generated by a match depends on both the customer type and the server. The customer types are latent and must be learned from reward observations. This additional uncertainty makes the resource-allocation problem substantially more difficult than in the homogeneous setting.

We consider a batched model, for example, a daily or weekly operating cycle, in which the platform processes a fixed panel of customers and then replenishes server capacities before processing the next batch. Each batch contains M customers, indexed by $m \in [M]$, and consists of N assignment rounds, indexed by $n \in [N]$. In every round, each customer generates exactly one task, so the platform processes M customer–task pairs per round and NM tasks over the batch. Equivalently, each customer contributes N tasks over the batch, which makes within-customer posterior learning possible. Customer m has latent type $z(m) \in [I]$, where

$$\mathbb{P}(z(m) = i) = \rho_i, \qquad i \in [I], \qquad \sum_{i=1}^{I} \rho_i = 1,$$

and the types are independent across customers. The platform manages J heterogeneous servers. For a fixed batch, server j has a known integer capacity $c_j \in \mathbb{N}$, with $c_j \ge 1$ for every $j \in [J]$ and $\sum_{j=1}^{J} c_j > NM$. We write $\mu_j = c_j/(NM)$ for its normalized capacity share, so $c_j = \mu_j NM$ and $\sum_{j=1}^{J} \mu_j > 1$. In statements that vary N or M, the vector $\boldsymbol{\mu}$ is held fixed and the parameters range

over compatible horizon–batch-size pairs for which $c_j = \mu_j NM \in \mathbb{N}$ for every j. If a task from a type-i customer is assigned to server j, the reward is Bernoulli with mean $r_{i,j}$, and the remaining capacity of server j decreases by one. Within each round, the M tasks are processed in the fixed order $m = 1, \ldots, M$; any deterministic order gives the same model and analysis. At the beginning of the next batch, the capacity of each server j is replenished to c_j. We assume that the platform does not know the type distribution $\{\rho_i\}_{i=1}^I$, the reward matrix $\{r_{i,j}\}_{i\in[I],j\in[J]}$, or the individual customer types.

The platform aims to maximize the expected total reward collected within a batch. Because M captures the scale of the batch, it is convenient to normalize by M and use the expected *reward rate per customer* as the performance metric. If the type distribution and the reward matrix were known, then the reward rate per customer would be upper bounded by the optimal value of the linear program

$$\begin{aligned} \psi^* = \max_{\{p_{i,j}\}} \quad & N \sum_{i=1}^{I} \sum_{j=1}^{J} \rho_i p_{i,j} r_{i,j} \\ \text{s.t.} \quad & \sum_{i=1}^{I} \rho_i p_{i,j} \le \mu_j, \qquad j \in [J], \\ & \sum_{j=1}^{J} p_{i,j} = 1, \quad i \in [I], \qquad p_{i,j} \ge 0, \quad i \in [I],\ j \in [J]. \end{aligned} \tag{3}$$

Here $p_{i,j}$ denotes the average fraction of tasks from a type-i customer that is assigned to server j. Suppose the platform knows the latent type space, the type proportions $\{\rho_i\}_{i=1}^I$, and the reward matrix $\{r_{i,j}\}_{i\in[I],j\in[J]}$, but does not observe each customer's realized latent type. Then the uncertainty lies entirely in the individual latent type $z(m)$, and the posterior update is exactly a customer-level latent-type learning problem under capacity constraints. This is the benchmark studied in Johari et al. (2021). Our heterogeneous model is strictly harder because neither the type proportions nor the reward matrix are assumed known a priori; instead, they must be estimated from data and then reused through informative priors across batches.

Unlike the homogeneous LP, problem (3) no longer admits a simple rank-order structure in Lemma 1. In particular, the optimal solution need not be characterized by filling servers in decreasing order of reward. This loss of structure is the main reason why the UCB analysis from the homogeneous model does not extend cleanly. More importantly, the latent customer types make the effective LP coefficients customer-specific and partially observed. Accordingly, we do not provide a full regret theorem for heterogeneous UCB variants in this paper. Instead, we focus the theory on a Thompson-sampling approach, which does not rely on a rank-order characterization and is more robust to changing or non-unique LP optima.

The homogeneous TS-PS algorithm already suggests the right template: sample rewards from the current posterior, solve an LP with the sampled rewards, use probability sampling to execute the LP solution, and invoke a feasibility-repair step only when needed. In Section 4.1, we study learning and allocation within a single batch. In Section 4.2, we describe how information from previous batches can be used to construct improved priors for future batches.

4.1 Resource Allocation within One Batch

Fix one batch of M customers. For customer $m \in [M]$, let $z(m)$ denote the latent type of the customer and define $\boldsymbol{r}(m) = (r_{z(m),1}, \ldots, r_{z(m),J})$. At assignment round $n \in [N]$, each of the M customers contributes one task. If customer m is assigned to server j at stage n, the realized reward is denoted by $X_j^n(m)$, where $X_j^n(m) \sim \text{Bernoulli}(r_j(m))$, independently across n, m, and j conditional on the latent reward vectors. Let $j_n(m) \in [J]$ denote the server selected for customer m at stage n. Each assignment may depend only on observations available before that decision, including assignments and rewards of customers processed earlier in the same stage, together with fresh auxiliary randomization independent of the latent reward vectors and all potential reward variables. For each customer m and server j, define $N_j^n(m) = \sum_{s=1}^{n} \mathbb{1}(j_s(m) = j)$ and $N_j^0(m) = 0$. We also let B_j denote the current remaining capacity of server j within the batch. An algorithm is admissible if every assignment is nonanticipating in this sense and it never assigns a task to a server with zero remaining capacity.

A direct analysis of (3) is inconvenient because the customer types are unobserved. In particular, learning the parameters ρ_i and $r_{i,j}$ together with the latent customer types would require a clustering step. However, clustering procedures can be sensitive to initialization and outliers, which is undesirable in the early stage of task assignment, when the platform is primarily exploring to learn customer information. We therefore reformulate the one-batch optimization problem at the *customer level*. If the reward vectors $\boldsymbol{r}(1), \ldots, \boldsymbol{r}(M)$ were known, the reward-rate benchmark would be

$$\begin{aligned} \psi(M) = \max_{\boldsymbol{p}(1),\ldots,\boldsymbol{p}(M)} \quad & \frac{N}{M} \sum_{m=1}^{M} \sum_{j=1}^{J} p_j(m) r_j(m) \\ \text{s.t.} \quad & \sum_{m=1}^{M} p_j(m) \le \mu_j M, \qquad j \in [J], \\ & \sum_{j=1}^{J} p_j(m) = 1, \quad m \in [M], \qquad p_j(m) \ge 0, \quad m \in [M],\ j \in [J]. \end{aligned} \tag{4}$$

Here $\boldsymbol{p}(m) = (p_1(m), \ldots, p_J(m))$ is the average allocation vector for the N tasks of customer m. This formulation implicitly aggregates the latent type distribution and the reward matrix through the realized collection of customer reward vectors. A similar idea is also used in the UCB algorithms of Hsu et al. (2022).

The next proposition shows that the customer-level (4) converges to the type-level benchmark (3) as the batch size grows.

Proposition 3. *It holds that $\psi(M) \to \psi^*$ almost surely, and $\mathbb{E}[\psi(M)] \to \psi^*$, as $M \to \infty$.*

The algorithm uses the following customer-level Bayesian model. For each customer m, we place a prior distribution $\mathbb{Q}^{1,m}$ on the unknown reward vector $\boldsymbol{r}(m)$. For the regret analysis, we assume the joint prior is the product $\bigotimes_{m=1}^{M} \mathbb{Q}^{1,m}$ and draw the vector of posterior samples from the corresponding joint posterior. This is consistent with the independent latent customer types and with the customer-level mixture priors used below. At the beginning of stage n, the platform

updates this prior to the posterior

$$\mathbb{Q}^{n,m} = \mathbb{Q}\left(\cdot \;\middle|\; j_1(m), X^1_{j_1(m)}(m), \ldots, j_{n-1}(m), X^{n-1}_{j_{n-1}(m)}(m)\right).$$

Thus, the posterior for customer m depends only on that customer's realized service history. In particular, if a sampled server is unavailable and the task is repaired to another server, the platform observes and uses only $(j_n(m), X^n_{j_n(m)}(m))$; the counterfactual reward from the unavailable sampled server is not observed.

Example 2 (Known type distribution and reward matrix, unknown latent type)**.** *Suppose the platform knows $\{\rho_i\}_{i=1}^I$ and $\{r_{i,j}\}_{i\in[I],j\in[J]}$, but does not observe the latent customer type. Then the uncertainty lies entirely in $z(m)$. For each $i \in [I]$, the prior mass on the reward vector $(r_{i,1}, \ldots, r_{i,J})$ is ρ_i. Define $W_j^{n-1}(m) = \sum_{s=1}^{n-1} X^s_{j_s(m)}(m)\mathbb{1}(j_s(m) = j)$. Then the posterior type probabilities satisfy*

$$\mathbb{Q}^{n,m}(z(m) = i) \propto \rho_i \prod_{j=1}^{J} r_{i,j}^{W_j^{n-1}(m)} (1 - r_{i,j})^{N_j^{n-1}(m) - W_j^{n-1}(m)}.$$

This is the latent-type setting studied in Johari et al. (2021).

Example 3 (Informative mixture-Beta prior)**.** *Suppose preliminary estimates v_i and $w_{i,j}$ of ρ_i and $r_{i,j}$ are available before the batch begins. Let $A_{i,j} = \lceil N w_{i,j} \rceil$ and $B_{i,j} = \lceil N(1 - w_{i,j}) \rceil$. A natural prior for customer m is then*

$$\mathbb{Q}^{1,m} = \sum_{i=1}^{I} v_i \bigotimes_{j=1}^{J} \mathrm{Beta}(A_{i,j}, B_{i,j}),$$

which is a mixture of product Beta distributions. Because each component is conjugate, the posterior remains a mixture of product Beta distributions:

$$\mathbb{Q}^{n,m} = \sum_{i=1}^{I} v_i^{n,m} \bigotimes_{j=1}^{J} \mathrm{Beta}\big(A_{i,j} + W_j^{n-1}(m), B_{i,j} + N_j^{n-1}(m) - W_j^{n-1}(m)\big),$$

where

$$v_i^{n,m} = \frac{v_i P_i^{n,m}}{\sum_{i'=1}^{I} v_{i'} P_{i'}^{n,m}} \quad \textit{and} \quad P_i^{n,m} = \prod_{j=1}^{J} \frac{\mathrm{B}\big(A_{i,j} + W_j^{n-1}(m), B_{i,j} + N_j^{n-1}(m) - W_j^{n-1}(m)\big)}{\mathrm{B}(A_{i,j}, B_{i,j})}.$$

Here $\mathrm{B}(\cdot,\cdot)$ denotes the Beta function.

Because the LP in the heterogeneous setting may admit multiple, rapidly changing optimal solutions, directly tracking the optimizer can prematurely deplete the capacity of certain servers. Consequently, we do not use direct tracking as the primary execution rule. Instead, we rely on probabilistic sampling and invoke tracking only as a feasibility-repair step when the selected server is unavailable. This leads to the following single-batch algorithm.

Lemma 5. *Algorithm 4 is admissible.*

Algorithm 4 TS-PS-Heterogeneous

Initialize $N_j^0(m) = 0$ for all $j \in [J]$ and $m \in [M]$, and $B_j = c_j$ for all $j \in [J]$.
for $n = 1, \ldots, N$ **do**
  Sample $\boldsymbol{S}^n(m) \sim \mathbb{Q}^{n,m}$ for all $m \in [M]$.
  Compute $(\boldsymbol{p}^n(1), \ldots, \boldsymbol{p}^n(M))$ as an optimizer of (4) with $r_j(m)$ replaced by $S_j^n(m)$.
  Independently sample $\widehat{j}_n(m) \sim \boldsymbol{p}^n(m)$ for all $m \in [M]$. ▷ Probability sampling
  for $m = 1, \ldots, M$ **do**
    if $B_{\widehat{j}_n(m)} > 0$ **then**
      Set $j_n(m) = \widehat{j}_n(m)$.
    else
      Set $j_n(m) \in \operatorname{argmax}_{j:B_j>0}\{np_j^n(m) - N_j^{n-1}(m)\}$. ▷ Tracking repair
    end if
    Observe $X^n_{j_n(m)}(m)$ and update $B_{j_n(m)} \leftarrow B_{j_n(m)} - 1$.
    Update $N_j^n(m) = N_j^{n-1}(m) + \mathbb{1}(j_n(m) = j)$ for all $j \in [J]$.
    Update the posterior for customer m: $\mathbb{Q}^{n,m} \mapsto \mathbb{Q}^{n+1,m}$ using $(j_n(m), X^n_{j_n(m)}(m))$.
  end for
end for

Remark 2 (Why CT, DT, and PS are implementable)**.** *The LP subroutines return fractional target allocations, whereas the platform must make one integral server assignment for every task. CT and DT implement a fractional target by maintaining only the current target vector, realized assignment counts, and remaining capacities; after the LP is solved, the server-selection step is an $O(J)$ comparison. CT tracks the accumulated sequence of targets and has a uniform discrepancy guarantee, while DT uses the current stagewise target and is simpler but can react strongly to rapidly changing or non-unique LP optimizers. PS requires one categorical draw from the LP vector and one capacity check; if the sampled server is unavailable, the DT repair selects an available server. Thus all three rules are online, require no future rewards or arrivals, and convert the LP recommendation into a pathwise-feasible assignment, as formalized by Lemmas 2, 3, 4, and 5.*

For fixed M, define

$$\mathrm{BayesRegret}_M(N) = \mathbb{E}\left[\psi(M) - \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{j_n(m)}(m)\right],$$

where the expectation is joint over the prior on the customer reward vectors, the reward realizations, and the algorithmic randomness. Theorem 4 gives the learning error for a fixed batch size M relative to the customer-level benchmark $\psi(M)$. When $M = 1$, this bound reduces to the homogeneous-case order in Theorem 3.

Theorem 4. *Under TS-PS-Heterogeneous with realized-assignment feedback,*

$$\mathrm{BayesRegret}_M(N) \le \left(6\sqrt{J} + 12\sqrt{J\log J}\right)\sqrt{N} + 3(J-1)\sqrt{\frac{JN}{M}}.$$

In particular, for fixed J and M, the one-batch Bayesian regret is $O(\sqrt{N})$.

Combining Theorem 4 with Proposition 3 yields the following asymptotic comparison with the type-level LP benchmark ψ^*.

Corollary 1. *Under TS-PS-Heterogeneous,*

$$\limsup_{M\to\infty} \mathbb{E}\left[\psi^* - \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{j_n(m)}(m)\right] \le \left(6\sqrt{J} + 12\sqrt{J\log J}\right)\sqrt{N}.$$

In particular, for fixed J, the asymptotic Bayesian regret rate remains $O(\sqrt{N})$.

Remark 3. *The parameter M captures the scale of a batch. Proposition 3 shows that larger batches provide a better approximation to the type-level benchmark ψ^*, while Theorem 4 shows that larger batches also reduce the feasibility-repair term $3(J-1)\sqrt{JN/M}$. On the other hand, the LP (4) becomes larger as M grows. Thus, increasing M improves statistical accuracy but also increases per-stage computational cost.*

4.2 Resource Allocation from Batch to Batch

For service delivered over successive batches, historical observations can be used to initialize future customer-level priors. The resulting clustering–modeling–assignment architecture is illustrated in Figure 1.

Let $\mathcal{D}^{\tau-1}$ denote the dataset collected from customers who completed service in the previous batches. Let H be the number of such customers. For each customer h and server j, let $n_j(h)$ be the number of tasks from customer h assigned to server j, and let $\alpha_j(h)$ be the corresponding reward collected by the platform. Thus,

$$\mathcal{D}^{\tau-1} = \{(n_j(h), \alpha_j(h)) : 1 \le j \le J,\ 1 \le h \le H\}.$$

This dataset serves as the input for batch τ. After the platform completes service in batch τ, the dataset is updated to $\mathcal{D}^{\tau}$, which is then used in the next batch. Within batch τ, the three stages are as follows:

1. **Clustering.** The platform uses $\mathcal{D}^{\tau-1}$ to estimate the unknown system parameters ρ_i and $r_{i,j}$.
2. **Modeling.** The platform converts the parameter estimates from the clustering stage into a prior model for ρ_i and $r_{i,j}$, together with the corresponding posterior update rule and sampling scheme; see Example 3.
3. **Assignment.** The platform uses the resulting prior model as input to Algorithm 4. After serving all customers in the current batch, it updates the dataset and passes it to the next batch.

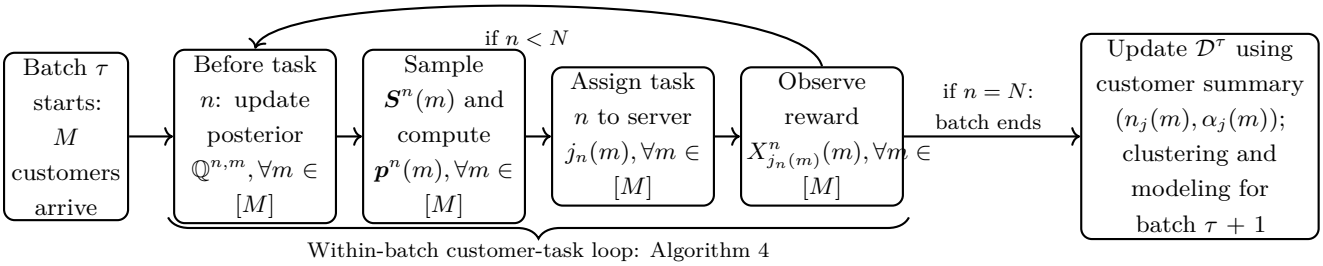

Figure 1: Within batch τ, we run Algorithm 4 with the initial posterior $\mathbb{Q}^{1,m}$ warm-started from the prior constructed using $\mathcal{D}^{\tau-1}$. For each task n, the platform samples $\boldsymbol{S}^n(m)$ from the current customer-specific posterior $\mathbb{Q}^{n,m}$, computes the assignment probabilities, assigns a server, observes the realized reward, and immediately updates the posterior to $\mathbb{Q}^{n+1,m}$. After the customer completes all N tasks, the resulting customer-level summary is recorded. These summaries are aggregated into $\mathcal{D}^\tau$, which is then used in the clustering and modeling stages to initialize batch $\tau + 1$.

Remark 4. *The latent-type formulation is not merely a richer parametrization of a pooled homogeneous mixture; it changes the information structure of the online problem. Because each customer contributes multiple tasks, observations from early tasks are informative about how the same customer should be matched later in the horizon. Under a latent-type prior, a success or failure on one server updates the posterior over the customer's hidden type and therefore changes the predicted rewards on all other servers simultaneously. This preserves the key mechanism needed for within-customer sequential adaptation, while also allowing statistical pooling across different customers who share the same latent type. By contrast, a pooled model averages the cross-server response patterns into a single reward vector. Its posterior update is then essentially serverwise: observing server j only revises the belief about server j, while beliefs about the remaining servers stay anchored at the same pooled average. The pooled model therefore discards exactly the cross-server information that should drive later assignments for that customer.*

Numerical experiments in Section 5.2.2 show that this is not merely a conceptual distinction. Our pooled warm-start baseline uses the same first-batch data, the same aEM estimates, the same prior strength, the same customer-level LP, and the same probability-sampling execution rule as TS-PS-1, and differs only in one respect: it replaces the learned individual latent label by a mixture of latent distributions. This seemingly mild simplification performs materially worse than the latent warm-start and, in our experiments, even worse than the cold-start policy TS-PS-0. The failure is therefore not due to a lack of warm-start information; it is caused by collapsing the learned latent structure into an "average customer" prior that is informative but systematically misspecified for sequential decision-making. In this setting, having a strong prior is not enough: the prior must preserve the latent structure that links a customer's responses across servers.

This interpretation is consistent with the broader literature. When the latent model is known, Johari et al. (2021) show that one can exploit this structure to obtain logarithmic regret. When the shared structure is unknown and must be estimated, bandit and dynamic-pricing papers likewise

report substantial gains from learning and reusing latent or clustered structure rather than treating users or products independently (Maillard and Mannor, 2014; Gentile et al., 2014, 2017; Miao et al., 2022; Pal et al., 2023). It also clarifies the role of our approach relative to Hsu et al. (2022): their algorithm integrates learning and control online, whereas our multi-batch clustering–modeling–assigning architecture is designed to estimate, refine, and reuse a common latent reward matrix and population proportions across batches. Once these population-level objects are learned with sufficient accuracy, the online problem in later batches becomes closer to identifying each customer's realized latent type than to relearning the entire reward matrix from scratch. Corollary 2 supports this view by establishing statistical consistency for the relevant latent population parameters.

To complete this architecture, it remains to specify how the platform estimates ρ_i and $r_{i,j}$ in the clustering stage. For each customer h, define $\boldsymbol{n}(h) = (n_1(h), \ldots, n_J(h))$ and $\boldsymbol{\alpha}(h) = (\alpha_1(h), \ldots, \alpha_J(h))$. Conditional on $z(h) = i$, the coordinates of $\boldsymbol{\alpha}(h)$ are independent binomial random variables, and the complete-data likelihood is

$$f(\boldsymbol{\alpha}(h), z(h) = i; \boldsymbol{\rho}, \boldsymbol{r}) = \rho_i \prod_{j=1}^{J} \binom{n_j(h)}{\alpha_j(h)} r_{i,j}^{\alpha_j(h)} (1 - r_{i,j})^{n_j(h) - \alpha_j(h)}.$$

Therefore, the marginal likelihood is

$$f(\boldsymbol{\alpha}(h); \boldsymbol{\rho}, \boldsymbol{r}) = \sum_{i=1}^{I} f(\boldsymbol{\alpha}(h), z(h) = i; \boldsymbol{\rho}, \boldsymbol{r}) = \sum_{i=1}^{I} \rho_i \prod_{j=1}^{J} \binom{n_j(h)}{\alpha_j(h)} r_{i,j}^{\alpha_j(h)} (1 - r_{i,j})^{n_j(h) - \alpha_j(h)},$$

and the log-likelihood is

$$\ell(\boldsymbol{\rho}, \boldsymbol{r}) = \sum_{h=1}^{H} \log \left(\sum_{i=1}^{I} \rho_i \prod_{j=1}^{J} \binom{n_j(h)}{\alpha_j(h)} r_{i,j}^{\alpha_j(h)} (1 - r_{i,j})^{n_j(h) - \alpha_j(h)} \right).$$

In principle, $(\boldsymbol{\rho}, \boldsymbol{r})$ can be estimated by maximum likelihood. Because of the summation inside the logarithm, it is natural to use the Expectation-Maximization (EM) algorithm.

4.2.1 EM algorithm

In the E-step, we compute the posterior responsibilities

$$w_{h,i}^{\text{old}} := \mathbb{P}\left(z(h) = i \mid \boldsymbol{\alpha}(h); \boldsymbol{\rho}^{\text{old}}, \boldsymbol{r}^{\text{old}} \right) \propto \rho_i^{\text{old}} \prod_{j=1}^{J} \binom{n_j(h)}{\alpha_j(h)} (r_{i,j}^{\text{old}})^{\alpha_j(h)} (1 - r_{i,j}^{\text{old}})^{n_j(h) - \alpha_j(h)}.$$

We then form

$$Q(\boldsymbol{\rho}, \boldsymbol{r}; \boldsymbol{\rho}^{\text{old}}, \boldsymbol{r}^{\text{old}}) = \sum_{h=1}^{H} \sum_{i=1}^{I} w_{h,i}^{\text{old}} \log f(\boldsymbol{\alpha}(h), z(h) = i; \boldsymbol{\rho}, \boldsymbol{r}).$$

In the M-step, we maximize Q with respect to $(\boldsymbol{\rho}, \boldsymbol{r})$, which yields the closed-form updates

$$\rho_i^{\text{new}} = \frac{1}{H} \sum_{h=1}^{H} w_{h,i}^{\text{old}}, \qquad r_{i,j}^{\text{new}} = \frac{\sum_{h=1}^{H} w_{h,i}^{\text{old}} \alpha_j(h)}{\sum_{h=1}^{H} w_{h,i}^{\text{old}} n_j(h)}.$$

Iterating the E-step and M-step yields the standard EM estimator.

4.2.2 Accelerated EM algorithm

Although standard EM is conceptually simple, it can converge slowly and can be sensitive to initialization. We therefore propose an accelerated EM procedure, abbreviated aEM, inspired by the EM updates above and the two-round EM idea in Dasgupta and Schulman (2007).

In addition to the dataset $\mathcal{D}^{\tau-1}$, aEM uses two tuning parameters: a threshold ρ_T and an over-clustering level L. The first round uses L initial centers; centers with sufficiently small mixing weights are then removed; finally, a second round is run on I representative centers selected by Farthest First Traversal.[3] The procedure is summarized in Algorithm 5. Under Assumptions 1 and 2, aEM achieves perfect cluster recovery after two rounds with high probability, so its output equals the empirical within-cluster averages. The formal statement is given in Proposition 4, and the proof is deferred to Appendix F.

Assumption 1 (Identifiability)**.** *No two rows of the reward matrix $\boldsymbol{r}$ are identical.*

Assumption 2 (Sufficient exploration)**.** *There exists a constant $a > 0$, specified in Appendix F, such that*

$$n_j(h) \geq a \log N, \qquad \forall h \in [H],\ \forall j \in [J].$$

Assumption 1 simply ensures that any two customer types are statistically identifiable. Assumption 2 ensures that each customer receives sufficient task service from each server so that the resulting clusters are well separated. One can enforce Assumption 2 by allocating $O(\log N)$ exploratory tasks uniformly across servers. We remark that the sufficient exploration condition is only needed for establishing consistency in our aEM algorithm. In practice, we observe that aEM can perform well even without strictly enforcing this condition. Indeed, in the numerical experiments, we never manually enforce it.

Recall that the latent customer types are i.i.d. with probabilities $\{\rho_i\}_{i=1}^I$, and let $\rho_{\min} = \min_{1\leq i\leq I} \rho_i$. Let $\mathcal{H}_i$ be the realized set of type-i customers in $\mathcal{D}^{\tau-1}$, and let $H_i = |\mathcal{H}_i|$.

Proposition 4. *Under Assumption 1 and Assumption 2, there exist instance-dependent constants $\underline{\rho}$ and $\overline{\rho}$ such that, for any $\rho_T \in (\underline{\rho}, \overline{\rho})$, with probability at least*

$$1 - \frac{2HJ}{N} - 2Ie^{-L\rho_{\min}/8} - 2Ie^{-H\rho_{\min}/48},$$

we have

$$\rho_i^{(2)} = \frac{H_i}{H}, \qquad r_{i,j}^{(2)} = \frac{1}{H_i} \sum_{h\in\mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)}. \tag{5}$$

Remark 5. *Although our proof uses tools related to Dasgupta and Schulman (2007), their two-round EM algorithm is not directly applicable here. Their analysis concerns mixtures of high-dimensional*

[3]The Farthest First Traversal of a bounded metric space generates a sequence of points in the space. The first point is selected by a fixed deterministic rule, and each successive point maximizes its distance from the previously selected set. Here we use Euclidean distance as the metric.

Algorithm 5 Accelerated Expectation-Maximization (aEM)

Input: ρ_T, L, $\mathcal{D} = \{n_j(h), \alpha_j(h) : 1 \le j \le J,\ 1 \le h \le H\}$.

Output: $\boldsymbol{\rho}^{(2)} = \{\rho_i^{(2)}\}_{i=1}^{I}$, $\boldsymbol{r}^{(2)} = \{r_{i,j}^{(2)}\}_{i=1,\dots,I;j=1,\dots,J}$.

1: *Initialization.* Randomly choose L customers $\{X(\ell) : 1 \le \ell \le L\}$ and set

$$\rho_\ell^{(0)} = \frac{1}{L}, \qquad \boldsymbol{r}_\ell^{(0)} = \left(\frac{\alpha_1(X(\ell))}{n_1(X(\ell))}, \dots, \frac{\alpha_J(X(\ell))}{n_J(X(\ell))}\right).$$

2: *First-round clustering.* For each customer h and center ℓ, compute

$$p_\ell^{(1)}(\boldsymbol{\alpha}(h)) \propto f(\boldsymbol{\alpha}(h), z(h) = \ell; \boldsymbol{\rho}^{(0)}, \boldsymbol{r}^{(0)}),$$

and update

$$\rho_\ell^{(1)} = \frac{1}{H}\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h)), \qquad r_{\ell,j}^{(1)} = \frac{\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h))\alpha_j(h)/n_j(h)}{\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h))}.$$

3: *Pruning.* Remove centers with $\rho_\ell^{(1)} < \rho_T$. Among the remaining centers, apply Farthest First Traversal under Euclidean distance to select I representative centers, and reset their mixing weights to

$$\widetilde{\rho}_i^{(1)} = \frac{1}{I}.$$

4: *Second-round clustering.* For each customer h, compute

$$p_i^{(2)}(\boldsymbol{\alpha}(h)) \propto f(\boldsymbol{\alpha}(h), z(h) = i; \widetilde{\boldsymbol{\rho}}^{(1)}, \boldsymbol{r}^{(1)}),$$

and define the MAP estimate

$$\widehat{z}(h) = \operatorname*{argmax}_{1 \le i \le I} p_i^{(2)}(\boldsymbol{\alpha}(h)).$$

Then update

$$\rho_i^{(2)} = \frac{1}{H}\sum_{h=1}^{H} \mathbb{1}\{\widehat{z}(h) = i\}, \qquad r_{i,j}^{(2)} = \frac{\sum_{h=1}^{H} \mathbb{1}\{\widehat{z}(h) = i\}\alpha_j(h)/n_j(h)}{\sum_{h=1}^{H} \mathbb{1}\{\widehat{z}(h) = i\}}.$$

spherical Gaussians with common variance, whereas our setting involves customer-level product-binomial observations with heterogeneous assignment counts $n_j(h)$. In our problem, cluster separation is driven by the number of tasks N, not by large ambient dimension.

The analysis in Appendix F shows that, for large N, choosing $\rho_T = 1/(3L)$ typically satisfies the condition $\underline{\rho} < \rho_T < \overline{\rho}$ in practice. See Lemma 25 and Lemma 26 for details. We recommend choosing $L = O(\log N)$. For fixed H, the terms $\frac{2HJ}{N}$ and $2Ie^{-L\rho_{\min}/8}$ in Proposition 4 vanish as N grows. The additional term $2Ie^{-H\rho_{\min}/48}$ reflects the concentration of the realized type frequencies H_i/H around the population proportions ρ_i. Therefore, for fixed H, the success probability in Proposition 4 approaches $1 - 2Ie^{-H\rho_{\min}/48}$ as $N \to \infty$. If H also grows and $H/N \to 0$, then the success probability approaches 1. We emphasize that Equation (5) is the sharp finite-sample estimate attainable after perfect clustering. Indeed, in finite samples the algorithm can recover only the realized type frequencies H_i/H and the realized within-cluster empirical averages. Corollary 2 shows that these finite-sample estimates inherit the concentration and consistency of the empirical type frequencies.

Corollary 2. *Under the assumptions of Proposition 4, for every $\epsilon > 0$, with probability at least*

$$1 - \frac{2HJ}{N} - 2Ie^{-L\rho_{\min}/8} - 2Ie^{-2H\epsilon^2} - Ie^{-H\rho_{\min}/8} - 2IJe^{-H\rho_{\min}\epsilon^2},$$

we have

$$\max_{1\le i\le I} |\rho_i^{(2)} - \rho_i| \le \epsilon \qquad \text{and} \qquad \max_{\substack{1\le i\le I\\ 1\le j\le J}} |r_{i,j}^{(2)} - r_{i,j}| \le \epsilon.$$

In particular, if $H \to \infty$, $N \to \infty$, $HJ/N \to 0$, and $L \to \infty$ (for example, $L = c\log N$ for any fixed $c > 0$), then

$$\max_{1\le i\le I} |\rho_i^{(2)} - \rho_i| \to 0, \qquad \max_{\substack{1\le i\le I\\ 1\le j\le J}} |r_{i,j}^{(2)} - r_{i,j}| \to 0$$

in probability.

Remark 6 (Scope of the current theory)**.** *The batchwise architecture is not accompanied by a multi-batch regret theorem. The formal results establish that latent types provide the appropriate state representation for within-customer adaptation and that, under accurate clustering, the population parameters (ρ, r) can be learned consistently from historical batches. A sharp multi-batch bound would additionally require control of clustering errors, prior misspecification, and across-batch dependence in the online policy.*

5 Numerical Experiments

The experiments use a common three-stage template: reward estimation, LP-based allocation optimization, and pathwise-feasible task assignment. This template covers UCB-DT (Algorithm 6), UCB-CT (Algorithm 2), UCB-PS, TS-PS (Algorithm 3), and TS-PS-Heterogeneous (Algorithm 4).[4]

[4]With a slight abuse of notation, TS-PS refers to both Algorithms 3 and 4.

Section 5.1 compares execution rules while holding the UCB estimation layer fixed, and Section 5.3 compares reward-estimation methods while using probability sampling for execution. The term *bandit learning* refers to reward estimates generated by a multi-armed-bandit algorithm.

5.1 Comparisons of Tracking and Probability Sampling

We compare three task-assignment rules, direct tracking (DT), cumulative tracking (CT), and probability sampling (PS), first in a homogeneous benchmark and then in a heterogeneous system. Recall that $j_n(m)$ denotes the server assigned to customer m in round n, and B_j^n is the remaining capacity of server j at the beginning of round n. To ensure admissibility, each rule restricts attention to feasible servers with $B_j^n > 0$. DT assigns customer m to a feasible server satisfying $j_n(m) \in \arg\max_{j:B_j^n>0}\{np_j^n(m) - N_j^{n-1}(m)\}$, that is, the server with the largest current tracking gap. CT instead tracks the cumulative target allocation and sets $j_n(m) \in \arg\max_{j:B_j^n>0}\{\sum_{n'=1}^{n} p_j^{n'}(m) - N_j^{n-1}(m)\}$. Finally, PS first samples $\widehat{j}_n(m) \sim \boldsymbol{p}^n(m)$; if $B^n_{\widehat{j}_n(m)} > 0$, then $j_n(m) = \widehat{j}_n(m)$, and otherwise the assignment falls back to the DT rule.

5.1.1 Homogeneous execution benchmark

We first compare the three UCB-based policies in a homogeneous system. UCB-DT is implemented exactly as Algorithm 6, and UCB-CT exactly as Algorithm 2. UCB-PS uses the UCB-DT estimation and LP layer, but implements the resulting fractional target by probability sampling with a feasible DT repair. Thus UCB-DT and UCB-PS isolate the execution rule, whereas UCB-CT also uses the residual-horizon normalization and horizon-based confidence radius required by its formal definition.

Example 4 (Ten servers and homogeneous tasks)**.** *We consider a platform with ten servers and capacity vector* $\boldsymbol{\mu} = (0.1, 0.1, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)$*. The tasks are homogeneous, and the reward vector is* $\boldsymbol{r} = (0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0.01)$*. Since rewards are decreasing in the server index, the optimal allocation exhausts the highest-reward servers until the total load reaches one. Hence* $\boldsymbol{p}^* = (0.1, 0.1, 0.3, 0.4, 0.1, 0, 0, 0, 0, 0)$*, and the LP upper bound on the expected average reward per task is* 0.67*.*

Figure 2 provides a homogeneous reference point for the heterogeneous comparison below. At $N = 50$, the mean terminal rewards are 0.407, 0.498, and 0.488 for UCB-DT, UCB-CT, and UCB-PS, respectively. At $N = 5000$, the corresponding values are 0.626, 0.640, and 0.640. CT and PS are therefore close at the short horizon and nearly indistinguishable at the long horizon. UCB-DT remains feasible and improves steadily, but finishes below the two policies that pace assignments through cumulative tracking or randomized execution. Because UCB-CT also uses its theorem-specific residual-horizon LP and confidence radius, this figure should be read as a comparison of the named policies rather than as a pure execution-rule ablation. Nevertheless, the reward paths do not show the abrupt late-horizon deterioration observed for UCB-DT in the heterogeneous example below. This supports the interpretation that the severe heterogeneous failure is driven by the unstable customer-level decomposition of a non-unique LP, rather than by direct tracking alone.

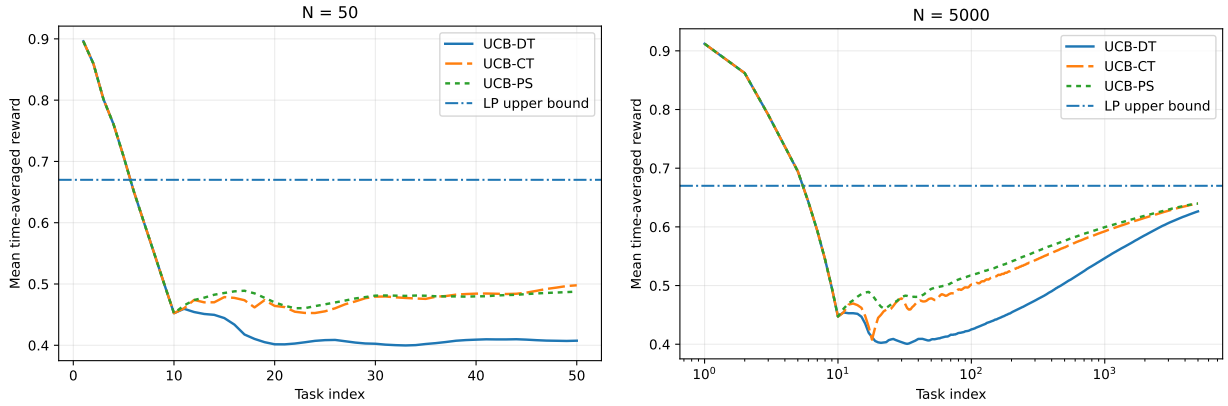

Figure 2: Mean time-averaged reward over 1000 common-random-number replications for UCB-DT, UCB-CT, and UCB-PS in Example 4, with $N = 50$ on the left and $N = 5000$ on the right. UCB-DT and UCB-CT follow Algorithms 6 and 2; UCB-PS uses the UCB-DT estimation/LP layer with probability sampling and DT repair. The dash-dotted line marks the LP upper bound 0.67.

5.1.2 Heterogeneous execution benchmark

The heterogeneous benchmark compares UCB-DT, UCB-CT, and UCB-PS in Example 5. To isolate the effect of the task-assignment sub-routine, we set each realized reward equal to the corresponding true mean reward.

Example 5 (Ten servers and three heterogeneous customer types)**.** *We consider a platform with ten servers and capacity vector* $\boldsymbol{\mu} = (0.15, 0.08, 0.14, 0.07, 0.16, 0.15, 0.10, 0.06, 0.12, 0.09)$*. The reward matrix* $\boldsymbol{r}$ *is*

$$\begin{matrix} \textit{type 1} \\ \textit{type 2} \\ \textit{type 3} \end{matrix} \begin{pmatrix} 0.67 & 0.39 & 0.08 & 0.59 & 0.14 & 0.69 & 0.16 & 0.86 & 0.25 & 0.86 \\ 0.79 & 0.93 & 0.36 & 0.14 & 0.57 & 0.69 & 0.28 & 0.63 & 0.57 & 0.69 \\ 0.38 & 0.19 & 0.73 & 0.23 & 0.08 & 0.28 & 0.08 & 0.21 & 0.88 & 0.12 \end{pmatrix}.$$

The entries are randomly generated. Each batch contains $M = 10$ *customers: five of type 1, three of type 2, and two of type 3. The corresponding type distribution is* $\boldsymbol{\rho} = (0.5, 0.3, 0.2)$*. Solving the upper-bound problem yields the optimal allocation*

$$\boldsymbol{p}^* = \begin{pmatrix} 0.26 & 0 & 0 & 0.14 & 0 & 0.3 & 0 & 0.12 & 0 & 0.18 \\ 0.0667 & 0.2667 & 0 & 0 & 0.5333 & 0 & 0 & 0 & 0.1333 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 \end{pmatrix}.$$

The resulting upper bound on the average reward is 0.7231*.*

Figure 3 plots the time-averaged reward over the $N = 5000$ task-assignment stages. All three policies rise quickly at the beginning and then gradually approach the upper bound, shown by the horizontal reference line. UCB-DT initially delivers the highest time-averaged reward, but its improvement essentially stops after Server 8 is depleted, and the curve declines after Server 10 is depleted. As a result, UCB-DT ends with a lower total reward over the horizon than UCB-CT and UCB-PS. By contrast, UCB-CT and UCB-PS remain nearly indistinguishable throughout the horizon and continue to improve until the end. The plot therefore shows that DT is more aggressive in the short run but less effective over the full horizon.

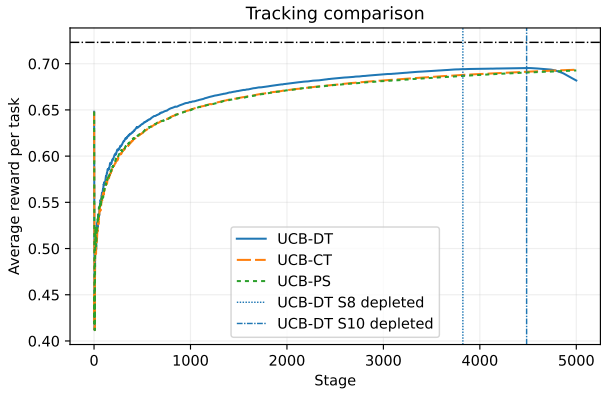

Figure 3: Evolution of the time-averaged reward under UCB-DT, UCB-CT, and UCB-PS, where the two vertical lines mark the depletion times of Servers 8 and 10 under UCB-DT.

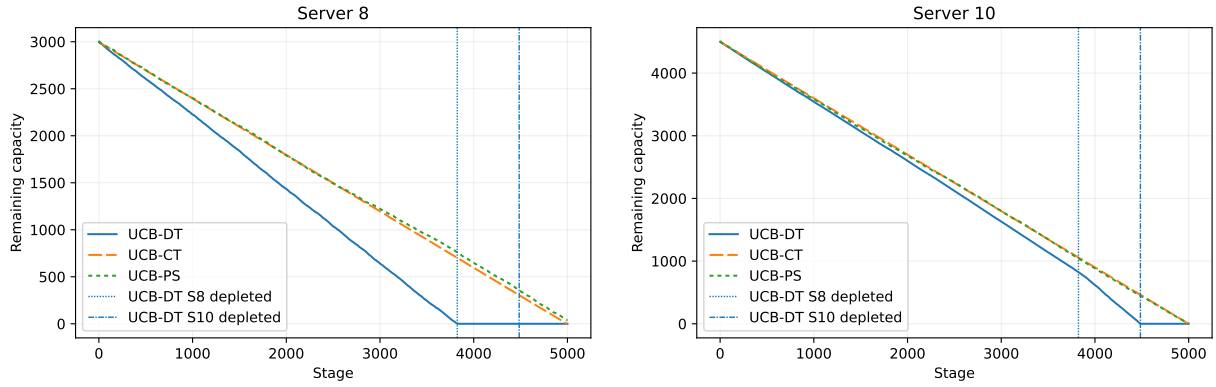

Figure 4: Remaining capacities of Servers 8 and 10 under UCB-DT, UCB-CT, and UCB-PS.

Figure 4 explains this difference through the remaining capacities of Servers 8 and 10. Under UCB-DT, Server 8 is depleted at about stage 3800, and Server 10 is depleted at about stage 4500. Under UCB-CT and UCB-PS, both servers remain available until close to the end of the horizon. Since Servers 8 and 10 are the most rewarding servers for type 1 customers and are used only by type 1 in the optimal allocation, their early depletion under UCB-DT forces later type 1 tasks to be assigned to less profitable servers. This explains the flattening after the first depletion time and the decline after the second. In contrast, CT and PS consume these key capacities more evenly, which yields a more stable reward trajectory and a higher total reward.

It is interesting that DT is highly effective in bandit problems such as best-arm identification (Garivier and Kaufmann, 2016) and structured regret minimization (Degenne et al., 2020), yet it can fail in the present online resource allocation problem. The key reason is that the customer-level LP (4) does not uniquely determine how the optimal type-level allocation should be decomposed across customers of the same type. In Example 5, recall that the type-level optimal solution of (3) is

$$\boldsymbol{p}^* = \begin{pmatrix} 0.26 & 0 & 0 & 0.14 & 0 & 0.3 & 0 & 0.12 & 0 & 0.18 \\ 0.0667 & 0.2667 & 0 & 0 & 0.5333 & 0 & 0 & 0 & 0.1333 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0 & 0 & 0 & 0.4 & 0 \end{pmatrix}.$$

If we index the five type 1 customers by $m = 1, \ldots, 5$, the three type 2 customers by $m = 6, \ldots, 8$, and the two type 3 customers by $m = 9, 10$, then any customer-level allocation matrix of the following form is optimal:

$$\boldsymbol{p}^* = \begin{pmatrix} a_1 & 0 & 0 & b_1 & 0 & s_1 & 0 & d_1 & 0 & e_1 \\ a_2 & 0 & 0 & b_2 & 0 & s_2 & 0 & d_2 & 0 & e_2 \\ a_3 & 0 & 0 & b_3 & 0 & s_3 & 0 & d_3 & 0 & e_3 \\ a_4 & 0 & 0 & b_4 & 0 & s_4 & 0 & d_4 & 0 & e_4 \\ a_5 & 0 & 0 & b_5 & 0 & s_5 & 0 & d_5 & 0 & e_5 \\ f_1 & g_1 & 0 & 0 & h_1 & 0 & 0 & 0 & i_1 & 0 \\ f_2 & g_2 & 0 & 0 & h_2 & 0 & 0 & 0 & i_2 & 0 \\ f_3 & g_3 & 0 & 0 & h_3 & 0 & 0 & 0 & i_3 & 0 \\ 0 & 0 & j_1 & 0 & 0 & 0 & 0 & 0 & k_1 & 0 \\ 0 & 0 & j_2 & 0 & 0 & 0 & 0 & 0 & k_2 & 0 \end{pmatrix}. \tag{6}$$

More precisely, if $\bar{a}, \bar{b}, \bar{s}, \bar{d}, \bar{e}$ denote the averages over the first five rows, $\bar{f}, \ldots, \bar{i}$ denote the averages over the next three rows, and $\bar{j}, \bar{k}$ denote the averages over the last two rows, then (6) is optimal whenever

$$\bar{a} = 0.26, \quad \bar{b} = 0.14, \quad \bar{s} = 0.3, \quad \bar{d} = 0.12, \quad \bar{e} = 0.18,$$

$$\bar{f} = 0.0667, \quad \bar{g} = 0.2667, \quad \bar{h} = 0.5333, \quad \bar{i} = 0.1333, \quad \bar{j} = 0.6, \quad \bar{k} = 0.4,$$

and each row sums to one. Although a symmetric optimizer exists, the LP may have many equivalent customer-level decompositions. The standing lexicographic rule selects one of them, but the selected optimizer can still change discontinuously as the estimated coefficients change. Therefore, even when the reward estimates stabilize, an adaptive algorithm that re-solves (4) at every stage may keep returning different customer-level optimizers. DT tracks this moving target too aggressively. Although every assignment remains feasible, DT can overuse scarce high-reward servers early and thus distort the intended pacing of capacity consumption. This is exactly what happens for Servers 8 and 10 in Figure 4.

To see why CT and PS avoid this issue, note that every stagewise LP solution satisfies $\sum_{m=1}^{M} p_j^n(m) \le \mu_j M$ for each server j. Summing over $n \le t$ yields $\sum_{m=1}^{M} \sum_{n=1}^{t} p_j^n(m) \le \mu_j t M$. Hence the LP prescribes a cumulative capacity budget for each server. For CT, Lemma 9 implies

that $|N_j^t(m) - \sum_{n=1}^t p_j^n(m)|$ is uniformly bounded for each pair (m, j) as long as Server j remains available. Summing over m shows that the realized usage of Server j stays close to its cumulative LP budget. For PS, $\mathbb{E}[N_j^t(m)] = \sum_{n=1}^t p_j^n(m)$, and martingale concentration implies that $N_j^t(m)$ stays close to this mean with high probability. Thus CT and PS pace capacity consumption much more evenly over time, which allows the time-averaged reward to keep improving throughout the horizon.

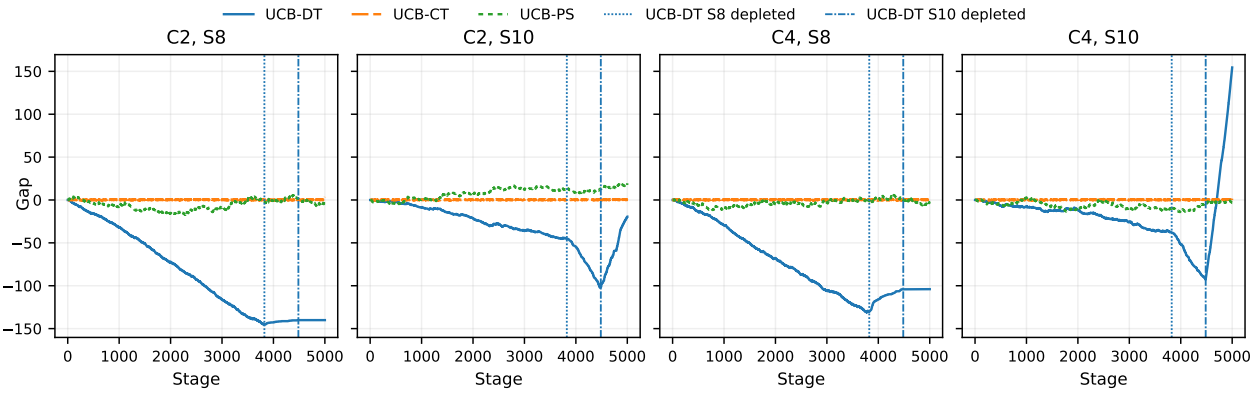

Figure 5: Evolution of the cumulative gap $\sum_{n=1}^t p_j^n(m) - N_j^t(m)$ for four representative customer-server pairs under UCB-DT, UCB-CT, and UCB-PS.

Figure 5 makes this mechanism explicit. The four panels correspond to two representative type 1 customers and the two critical servers, 8 and 10. Under UCB-DT, the cumulative gaps drift far away from zero. For $(C2, S8)$ and $(C4, S8)$, the gap becomes strongly negative before stage 3800, which shows that Server 8 is overused early relative to the cumulative LP prescription. A similar pattern appears for Server 10 before its depletion time. After Server 10 is depleted, the gap for $(C4, S10)$ turns sharply positive because the LP continues to assign mass to Server 10 while the realized assignment can no longer follow. By contrast, under UCB-CT the gaps stay essentially at zero throughout the horizon, and under UCB-PS they fluctuate around zero with only modest stochastic variation. This confirms that CT and PS track the cumulative LP prescription, whereas DT chases a moving stagewise optimizer. This pathology is specific to the heterogeneous setting. In the homogeneous setting, DT does not exhibit this behavior and still converges to the optimal allocation.

5.2 Benefits of Latent-Type Estimation

In this section, we highlight the clear advantages of customer-type inference and informative priors in the multi-batch learning algorithms. Throughout this subsection, we use the problem instance described in Example 5.

5.2.1 Batchwise Prior Refinement for TS-PS-τ

In this section, we use PS throughout and focus on how the prior quality evolves across batches. We consider three benchmark priors. TS-PS-0 uses independent Beta$(1,1)$ priors and represents a cold-start platform with little prior information. TS-PS-1 uses an informative prior constructed from estimated system parameters as in Example 3. TS-PS-∞ is an oracle benchmark that constructs the prior from the true system parameters as in Example 2. More generally, TS-PS-τ denotes the policy deployed in batch τ under the three-stage architecture.

The construction of TS-PS-τ is recursive. Batch 0 is served by TS-PS-0. After batch τ is completed, the platform augments the historical dataset with the newly collected service data and re-estimates the system parameters from the enlarged sample. For homogeneous tasks, the prior is updated from the sample mean rewards. For heterogeneous customers, we apply the accelerated EM algorithm (aEM, Algorithm 5) to the accumulated service data. Specifically, aEM first initializes L tentative centers from the data and performs a first-round over-clustering. We set $L = 2 \cdot n_{\text{types}} = 6$ in this experiment. It then prunes centers with small estimated mixing weights, selects I representative centers by the Farthest First Traversal rule, reassigns each customer to the most likely type by a MAP rule, and updates the type proportions and reward estimates. The resulting estimates are converted into the prior for the next batch, and TS-PS is then run with this updated prior. In particular, TS-PS-1 is the second-batch policy obtained after one warm-up batch of TS-PS-0.

In the experiments below, we do not enforce the sufficient condition $n_j(h) \geq a \log N$ in Assumption 2, and we do not add a separate forced-exploration phase for clustering. Instead, aEM is applied directly to the service data generated endogenously by TS-PS. Thus, the clustering stage does not receive an extra exploration budget beyond the natural exploration of Thompson sampling. Therefore, the gains observed below cannot be attributed to an artificial exploration subsidy for clustering. This makes the experiment informative about the practical cost of clustering. In particular, it shows that even without the forced $O(\log N)$ exploration required by the theory, the learned prior still yields respectable performance in this example.

Figure 6 shows that the three-stage architecture substantially improves the typical performance over the cold-start benchmark TS-PS-0. Moving from TS-PS-0 to TS-PS-1, both the median and the mean increase markedly, which shows the value of converting one batch of service data into an informative prior. At the same time, TS-PS-1 exhibits the most pronounced lower tail. This indicates that one batch of data is enough to improve the central performance, but not enough to stabilize the learned prior across all runs. In this example, the poor runs are caused by finite-sample estimation error and by occasional inaccuracies in the aEM step when no forced exploration is imposed.

As τ increases from 1 to 2, 4, 8, and 16, the lower tail becomes progressively shorter and the box plots move upward. Hence the main benefit of repeated batches is not only a higher average reward, but also a substantial reduction in downside risk. The improvement from $\tau = 0$ to $\tau = 1$ is the largest in terms of central tendency, whereas the later batches mainly refine the prior and suppress bad runs. By $\tau = 8$ and $\tau = 16$, the distribution of TS-PS-τ is already close to that of the

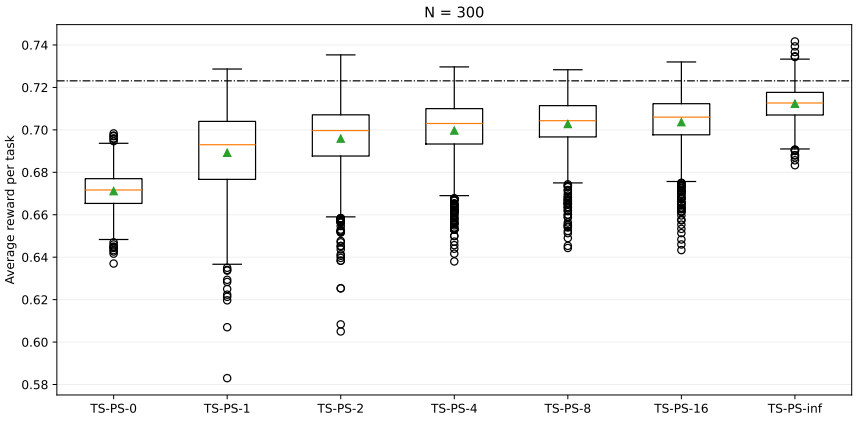

Figure 6: Box plots of the average reward per task under TS-PS-τ and TS-PS-∞ for Example 5. The horizon is $N = 300$ and $\tau \in \{0, 1, 2, 4, 8, 16\}$. The dash-dotted line marks the upper bound 0.7231. Results are based on 1000 runs.

oracle benchmark TS-PS-∞, although a small gap remains. This remaining gap is expected because even with a highly informative prior, the latent customer types are still unknown at arrival and must be learned online within each batch.

Overall, Figure 6 suggests that, in practice, the cost of clustering is more than offset by the improvement in the quality of the learned prior. Even without enforcing the $a \log N$ sampling condition required by the aEM theory, the three-stage architecture still delivers clear batch-to-batch gains and rapidly approaches the oracle benchmark. At least in this example, the sufficient condition in the aEM analysis appears conservative, and clustering remains effective even without additional forced exploration.

5.2.2 A pooled warm-start baseline

A natural question is whether the gain of TS-PS-1 comes simply from starting with an informative prior or from preserving the latent-type structure in that prior. To isolate this issue, we introduce a pooled warm-start baseline that uses exactly the same first-batch data, the same customer-level LP, and the same probability-sampling execution rule as TS-PS-1, but collapses the learned individual latent labels into a mixture of latent distributions.

Specifically, after the warm-up batch served by TS-PS-0, we apply the same aEM procedure and obtain preliminary estimates v_i and $w_{i,j}$ as in Example 3. The latent policy TS-PS-1 uses the mixture-Beta prior in Example 3. The pooled baseline instead forms the averaged reward vector $\bar{w}_j = \sum_{i=1}^{I} v_i w_{i,j}$ for $j \in [J]$, and defines

$$\widetilde{A}_j = \lceil N\bar{w}_j \rceil, \qquad \widetilde{B}_j = \lceil N(1-\bar{w}_j) \rceil, \quad \widetilde{\mathbb{Q}}^{1,m} = \bigotimes_{j=1}^{J} \mathrm{Beta}(\widetilde{A}_j, \widetilde{B}_j).$$

For customer m, its posterior after $n-1$ rounds is

$$\widetilde{\mathbb{Q}}^{n,m} = \bigotimes_{j=1}^{J} \mathrm{Beta}\big(\widetilde{A}_j + W_j^{n-1}(m), \widetilde{B}_j + N_j^{n-1}(m) - W_j^{n-1}(m)\big).$$

We denote the resulting policy by TS-PS-Pooled-1. This is a deliberately strong non-latent benchmark: it receives the same warm-start data and the same prior strength as TS-PS-1, and differs only in whether the learned prior retains latent customer type.

The difference in performance is therefore informative. Under the latent prior, an early success or failure on one server changes the posterior mass on the customer types and hence changes the predicted rewards on all other servers for that same customer. Under the pooled prior, the update is serverwise: observing server j only changes the Beta factor for server j, while the beliefs about the unobserved servers remain centered around the same pooled average. In other words, TS-PS-Pooled-1 still learns, but it cannot convert an early observation into cross-server inference about the customer's latent type. This distinction is crucial here because each customer contributes multiple tasks. Once early tasks reveal that a customer is more likely to be of one type than another, the later tasks should be redirected toward the servers favored by that type. The pooled model discards exactly this mechanism and instead treats every customer as an "average" customer.

This structural loss is clearly visible in the ten-server heterogeneous example (top panel of Figure 7). Over 1000 replications, the mean reward per task of TS-PS-Pooled-1 is 0.539, 0.569, 0.644, and 0.665 for $N \in \{50, 100, 200, 400\}$, respectively. The corresponding values for TS-PS-1 are 0.626, 0.665, 0.708, and 0.715, while the oracle TS-PS-∞ achieves 0.672, 0.695, 0.716, and 0.719. Notably, the pooled warm-start is even worse than the cold-start policy TS-PS-0, whose mean rewards are 0.584, 0.625, 0.686, and 0.700. Thus, in this example, simply warm-starting with a pooled prior does not help; it injects a confident but misspecified average reward vector that suppresses the customer-level adaptation needed later in the batch.

The same conclusion holds in the random heterogeneous examples (bottom panel, Figure 7). For $M = 10$ and $N = 50, 100, 200, 400$, the mean performance ratio of TS-PS-Pooled-1 is 0.830, 0.864, 0.896, and 0.920, respectively, compared with 0.913, 0.945, 0.967, and 0.980 for TS-PS-1 and 0.945, 0.968, 0.981, and 0.991 for TS-PS-∞. Again, TS-PS-Pooled-1 also falls below the cold-start benchmark TS-PS-0, whose mean performance ratios are 0.885, 0.922, 0.948, and 0.967. The message is therefore not merely that informative priors are useful. Rather, the prior must preserve the latent structure that links a customer's responses across servers. Collapsing the learned matrix $\{w_{i,j}\}$ into a single pooled vector removes precisely the information that makes within-customer sequential adaptation possible.

5.3 Comparison of Bandit Learning Algorithms

In this section, we fix PS as the task-assignment subroutine because of its strong theoretical guarantees and robust empirical performance. We then compare several bandit learning algorithms for the reward-estimation subroutine.

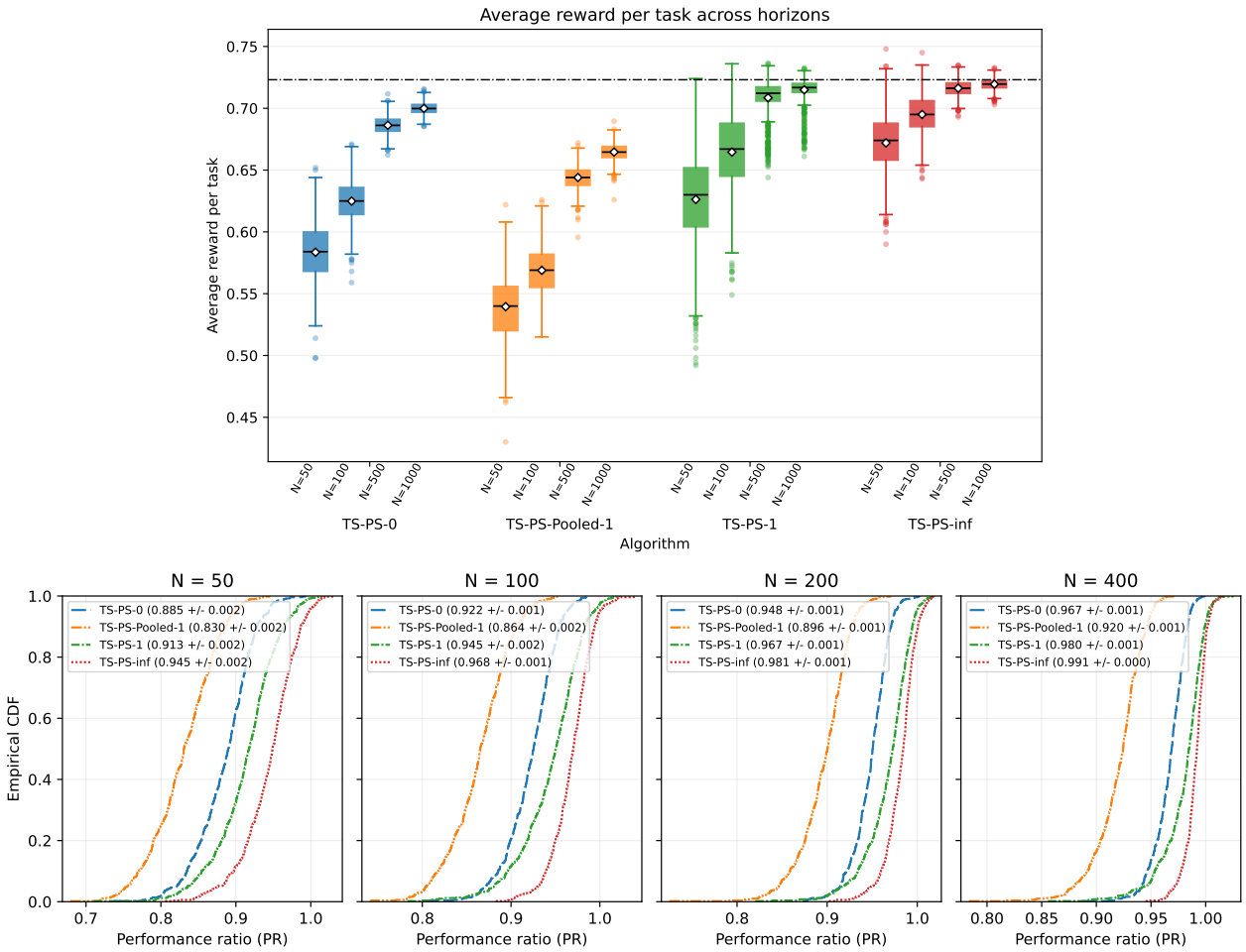

Figure 7: Comparison of pooled and latent approaches.

In addition to UCB-PS and TS-PS, we include a myopic benchmark, myopic-PS, which estimates rewards by sample means and performs no explicit exploration. A key feature of the allocation-proportion optimization subroutine is that the LP solution often lies at an extreme point, so the resulting allocation matrix is typically sparse, as illustrated by Example 5 and (6). As a result, some servers may receive zero allocation under the current estimate, which prevents their rewards from being updated accurately and can trap the algorithm in a suboptimal solution. We expect UCB and TS to mitigate this issue through systematic exploration.

For TS-PS, we consider three types of priors. TS-PS-0 uses independent Beta$(1, 1)$ priors and represents a cold-start platform with little prior information. TS-PS-1 uses an informative prior constructed from estimated system parameters, as in Example 3. TS-PS-∞ is an oracle benchmark that uses the true system parameters to construct the prior. See Section 5.2.1 for details.

For the homogeneous problem, we include a generic-BwK benchmark, which can be viewed as a specialization of UcbBwK (Badanidiyuru et al., 2018) to our setting. In each period, the algorithm forms optimistic reward estimates by augmenting each arm's empirical mean with a UCB bonus, then re-solves a *residual-budget linear program* using the remaining capacities and time-to-go to obtain a feasible sampling distribution over arms. It samples an arm from this distribution and updates both the reward estimates and the residual budgets.

5.3.1 Example: Ten Servers and Homogeneous Tasks

We return to Example 4 and compare generic-BwK, myopic-PS, UCB-PS, TS-PS-0, TS-PS-1, and TS-PS-∞. To facilitate a fair pathwise comparison, all algorithms are evaluated on the same pre-generated reward realizations: we pre-generate all reward draws before running any algorithm and then reveal them in the same order across algorithms, so the displayed differences are entirely driven by the assignment rules.

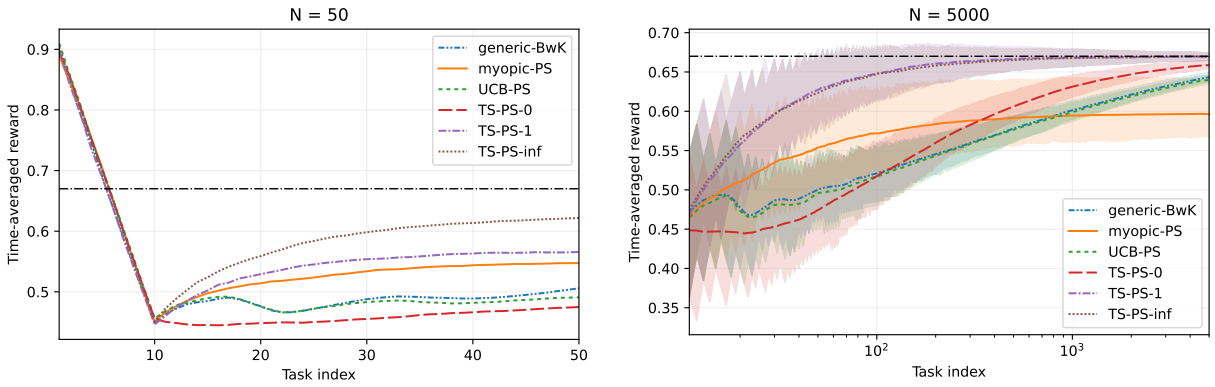

Figure 8: Average time-averaged reward over 1000 runs for Example 4, with $N = 50$ on the left and $N = 5000$ on the right. The dash-dotted line marks the upper bound 0.67. The shaded bands in the right panel show the first and third quartiles.

Figure 8 reports the mean time-averaged reward. For $N = 50$, TS-PS-∞ is clearly best and

TS-PS-1 is second. Myopic-PS performs well at this very short horizon, but generic-BwK already improves on UCB-PS and also on TS-PS-0, showing that residual-budget re-solving helps immediately. For $N = 5000$, the ranking is much sharper. TS-PS-1 and TS-PS-∞ both converge essentially to the upper bound 0.67, TS-PS-0 is clearly next, and generic-BwK finishes above both UCB-PS and myopic-PS. By the end of the horizon, the mean paths are approximately 0.669 for TS-PS-1, 0.669 for TS-PS-∞, 0.658 for TS-PS-0, 0.643 for generic-BwK, 0.640 for UCB-PS, and 0.596 for myopic-PS. Thus, re-solving against remaining capacity is materially better than the static UCB-PS benchmark, but it still falls well short of Thompson sampling.

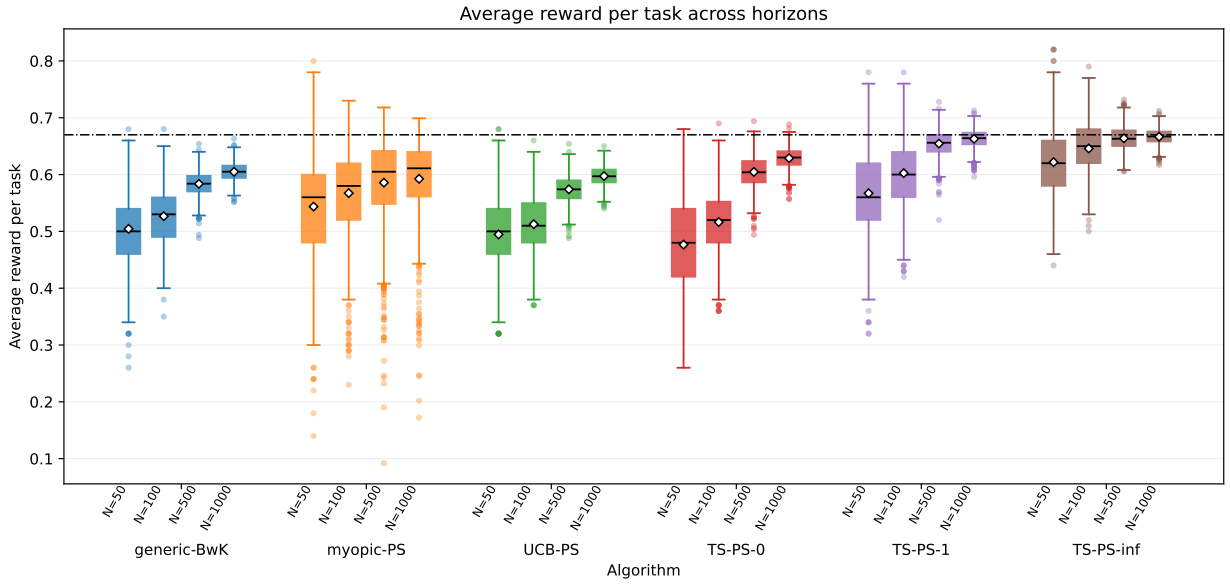

Figure 9: Box plots of the average reward per task over 1000 runs for Example 4, with $N \in \{50, 100, 500, 1000\}$. The dash-dotted line marks the upper bound 0.67.

Figure 9 confirms the same picture across $N \in \{50, 100, 500, 1000\}$. The comparison between generic-BwK and UCB-PS is especially clean: generic-BwK dominates UCB-PS at every horizon, with mean rewards 0.504 versus 0.495 at $N = 50$, 0.527 versus 0.513 at $N = 100$, 0.584 versus 0.574 at $N = 500$, and 0.605 versus 0.597 at $N = 1000$. It also has slightly smaller dispersion throughout. Relative to TS-PS-0, generic-BwK is stronger only at short horizons: it is better at $N = 50$ and $N = 100$, but TS-PS-0 overtakes it by $N = 500$ and remains clearly ahead at $N = 1000$. Relative to myopic-PS, generic-BwK is much more robust. Myopic-PS can have competitive medians, but it has by far the heaviest downside tail, with minimum rewards 0.14, 0.23, 0.092, and 0.172 across the four horizons. This downside risk keeps its mean well below its median and eventually pushes it below generic-BwK by $N = 1000$. Finally, Thompson sampling remains decisively strongest. The key advantage of an informative prior is that it pays off immediately, before there is enough time to learn from data alone. Relative to TS-PS-0, TS-PS-1 raises the mean reward from 0.477 to 0.567 at $N = 50$ and from 0.516 to 0.603 at $N = 100$, while also improving the lower tail substantially; for example, at $N = 100$ the first quartile increases from 0.48 to 0.56. Put differently, TS-PS-1

already closes about two-thirds of the gap between TS-PS-0 and the oracle TS-PS-∞ by $N = 100$. This is exactly where informative priors matter most: they reduce costly early exploration and steer capacity toward the right high-reward servers from the outset. As the horizon grows and data accumulate, the gap narrows but does not disappear: TS-PS-1 still improves on TS-PS-0 by about 0.050 at $N = 500$ and 0.034 at $N = 1000$, and by $N = 1000$ it closes more than 90% of the remaining gap to TS-PS-∞. Accordingly, TS-PS-1 is the best non-oracle method at every boxplot horizon, while TS-PS-∞ remains best overall. By $N = 1000$, their mean rewards are 0.663 and 0.667, both essentially at the benchmark 0.67.

5.3.2 Example: Ten Servers and Heterogeneous Customers and Tasks

We continue with Example 5. Figure 10 reports the mean time-averaged reward over 1000 runs, and Figure 11 reports the corresponding box plots across multiple horizons.

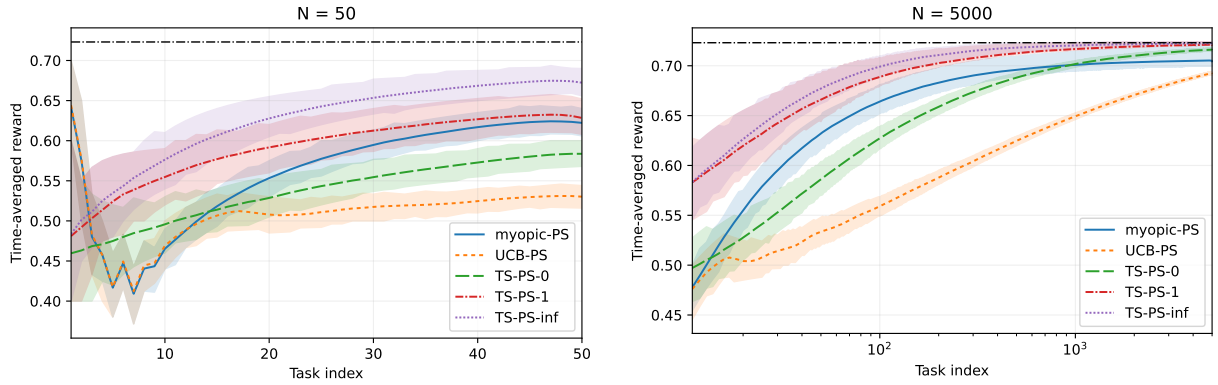

Figure 10: Mean time-averaged reward paths over 1000 runs for Example 5, with $N = 50$ on the left and $N = 5000$ on the right. The shaded bands indicate the first and third quartiles.

Figure 10 reveals a clear separation among the algorithms. When $N = 50$, myopic-PS is competitive because it does not pay an explicit exploration cost, but TS-PS-1 and especially TS-PS-∞ already perform better on average. By the end of the horizon, the mean time-averaged rewards are about 0.628 for TS-PS-1 and 0.672 for TS-PS-∞, compared with 0.622 for myopic-PS, 0.585 for TS-PS-0, and 0.530 for UCB-PS. When $N = 5000$, the separation becomes much clearer. TS-PS-1 and TS-PS-∞ nearly overlap and finish around 0.720 and 0.722, respectively, while TS-PS-0 reaches about 0.716. By contrast, myopic-PS plateaus around 0.705, and UCB-PS remains substantially lower at about 0.692. Thus, informative priors are valuable even at short horizons, and Thompson-sampling-based exploration becomes the most effective strategy as the horizon grows.

A notable feature of Figure 11 is that TS-PS-0 consistently outperforms UCB-PS at every horizon. The mean reward gap is about 0.055 at $N = 50$, 0.069 at $N = 100$, 0.064 at $N = 500$, and 0.053 at $N = 1000$. This shows that in heterogeneous systems the main issue is not only whether the algorithm explores, but also how it explores. As discussed in Section 4, UCB-style optimism is less effective when the optimal allocation does not admit a simple ranking structure. It tends to

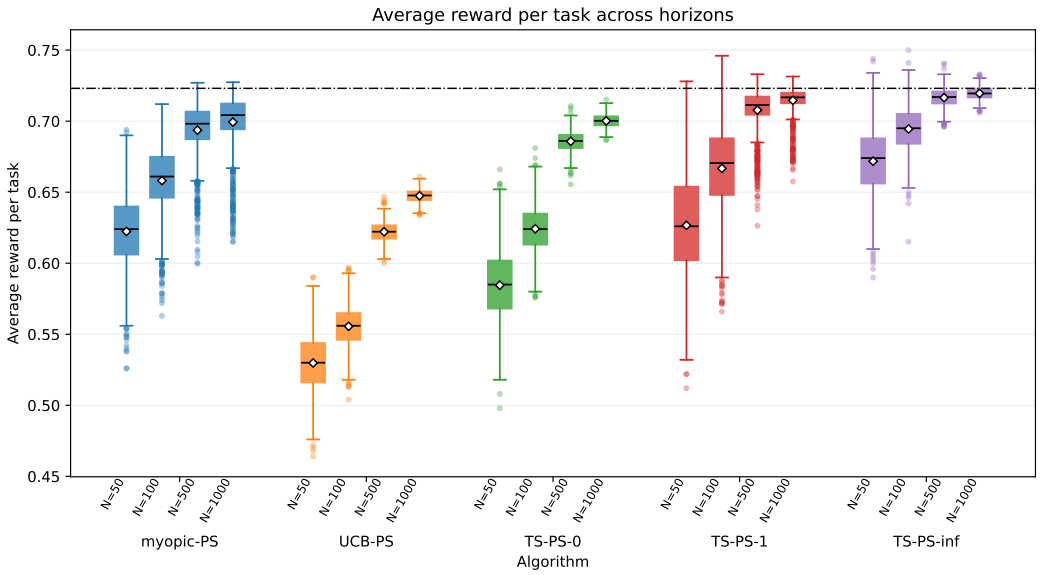

Figure 11: Box plots of the average reward per task over 1000 runs for Example 5, across horizons $N \in \{50, 100, 500, 1000\}$.

favor currently optimistic servers rather than the customer-server pairs that are most informative for resolving heterogeneous uncertainty. In contrast, Thompson sampling randomizes according to the posterior distribution and therefore generates a more balanced exploration pattern.

The box plots also show that myopic-PS has a pronounced lower tail, although its median can be competitive. This is consistent with the wide interquartile band in the right panel of Figure 10. Pure exploitation can work well in favorable sample paths, but it can also become trapped in seriously suboptimal allocations.

TS-PS-1 also exhibits more dispersion than the oracle benchmark TS-PS-∞, although the effect is much milder than for myopic-PS. At $N = 1000$, TS-PS-1 attains a mean reward of 0.715, close to the oracle value 0.720, but its minimum is only 0.658, compared with 0.706 for TS-PS-∞. This residual downside risk is caused by prior misspecification from finite first-batch data and, in the heterogeneous setting, by estimation error in the clustering stage. Because aEM is run on finite data, it can occasionally return imperfect clusters and hence a suboptimal informative prior. Nevertheless, the gap between TS-PS-1 and TS-PS-∞ becomes small as N grows, and Section 5.2.1 shows that repeated batchwise updating further reduces this gap.

Taken together, Figures 8 and 10 also clarify why TS performs strongly in the simulations without contradicting the regret statements. The $O(\log N)$ guarantees for UCB-CT and UCB-DT are instance-dependent: their constants scale inversely with the relevant sub-optimality gaps, and the number of exploratory assignments can scale as $\log N/\Delta^2$ when a critical gap is Δ. In Example 4, the reward gap at the LP cutoff is only 0.1, so these constants are material at the displayed horizons. Conversely, Theorem 3 is a worst-case Bayesian upper bound, not a claim that TS must incur $\sqrt{N}$ regret on every fixed instance; favorable priors and well-separated posterior mass can yield much

smaller realized regret. TS-PS-1 and TS-PS-∞ additionally benefit from informative priors, which reduce early exploration before capacity has been committed. Thus the plots compare finite-horizon constants and prior information, whereas the theorems describe different asymptotic and worst-case regimes.

5.3.3 Randomly Generated Examples

We next consider randomly generated heterogeneous instances, following a setup similar to Johari et al. (2021). We generate 500 examples with three customer types and five server types. In each instance, the customer-type distribution is fixed at $\boldsymbol{\rho} = (0.5, 0.3, 0.2)$. For each server type j, the normalized capacity share μ_j is drawn independently from the uniform distribution on $[1/10, 3/10]$. If $\sum_{j=1}^{J} \mu_j \leq 1$, we resample until the total capacity exceeds one. For each customer-server pair, the mean reward is drawn independently from the uniform distribution on $[0, 1]$. As in Johari et al. (2021), we work in a static setting in which each batch contains exactly $M\rho_i$ customers of type i. We consider two batch sizes, $M \in \{10, 10^3\}$, and horizons $N \in \{50, 100, 200, 400\}$. Thus, each batch contains $(5, 3, 2)$ customers of the three types when $M = 10$, and $(500, 300, 200)$ when $M = 10^3$. For each instance, the benchmark is the optimal value ψ^* of problem (3). Following Johari et al. (2021), we evaluate each algorithm by its performance ratio (PR), defined as the achieved average reward per customer per task divided by ψ^*/N. For each pair (M, N), we average the PR over the 500 instances.

Figure 12 reports the empirical cumulative distribution functions of the PR over the 500 random examples. Several patterns are robust across both batch sizes. First, TS-PS-∞ is uniformly best. Its ECDF is the rightmost curve in every panel, and its mean PR increases from 0.943 to 0.990 when $M = 10$ and from 0.950 to 0.992 when $M = 10^3$ as N grows from 50 to 400. Second, TS-PS-0 consistently dominates UCB-PS at every horizon and both batch sizes, showing again that in heterogeneous systems the form of exploration is crucial. Third, all algorithms improve as N increases, but the magnitude of the improvement depends strongly on how efficiently the algorithm uses the additional data.

The $M = 10^3$ experiment sharpens these conclusions. Relative to the top panel, the ECDFs in the bottom panel are uniformly shifted to the right and are visibly steeper. This is because a larger batch both provides more reward observations per stage and makes the realized assignment closer to the fluid LP prescription. The gain is especially pronounced for TS-PS-1. At $N = 50$, its mean PR increases from 0.912 when $M = 10$ to 0.945 when $M = 10^3$. By contrast, the corresponding increase is only from 0.943 to 0.950 for TS-PS-∞, from 0.915 to 0.923 for myopic-PS, from 0.887 to 0.895 for TS-PS-0, and from 0.822 to 0.825 for UCB-PS. Hence the main beneficiary of the larger batch is the warm-started TS approach that relies on a learned prior.

This comparison also changes the relative ranking between TS-PS-1 and myopic-PS. When $M = 10$, myopic-PS is slightly better than TS-PS-1 at $N = 50$, with mean PRs 0.915 and 0.912, but TS-PS-1 overtakes it from $N = 100$ onward. When $M = 10^3$, TS-PS-1 dominates myopic-PS already at $N = 50$, with mean PRs 0.945 and 0.923, and this advantage persists at all larger horizons. The

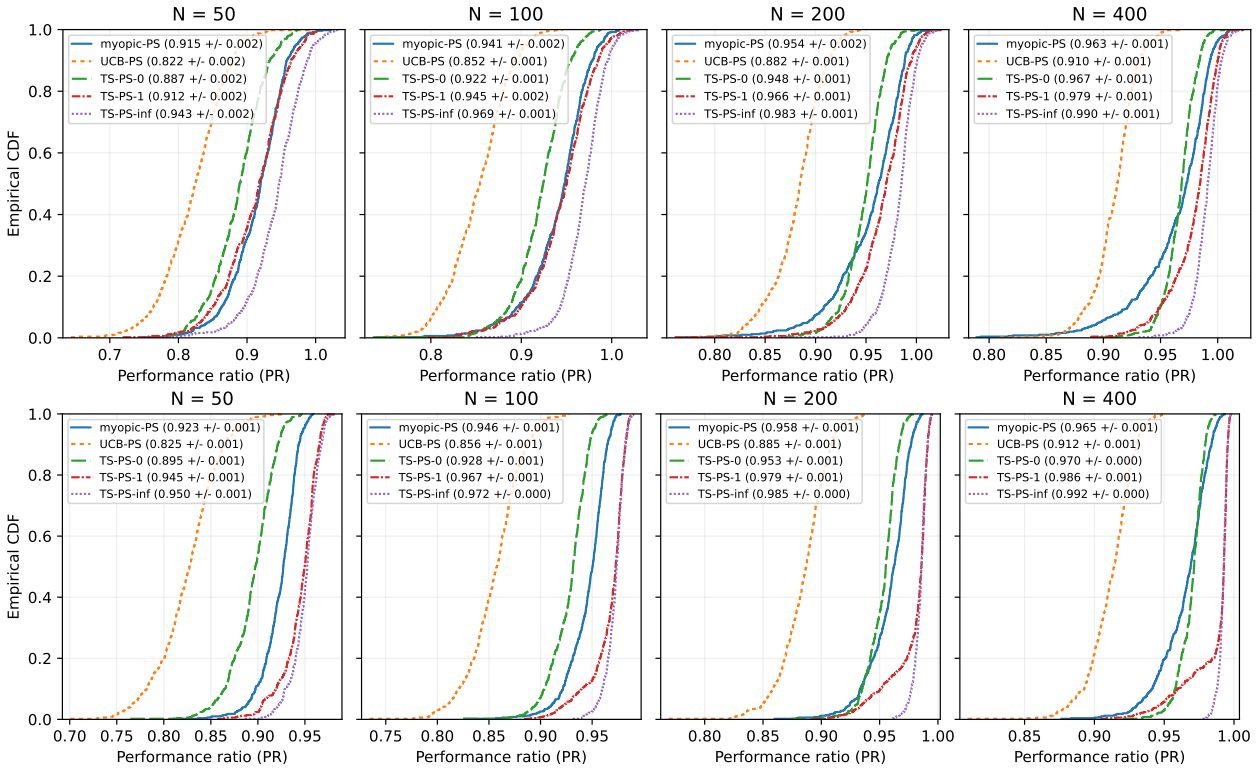

Figure 12: Empirical cumulative distribution functions of the performance ratio over 500 random examples under different algorithms for $N \in \{50, 100, 200, 400\}$, with $M = 10$ in the top panel and $M = 10^3$ in the bottom panel. The legends report the sample mean and standard error.

reason is that the larger first batch provides much more service data for estimation and clustering, so the informative prior used by TS-PS-1 becomes substantially more accurate. Indeed, the mean gap between TS-PS-1 and the oracle benchmark TS-PS-∞ is 0.031 at $(M, N) = (10, 50)$, but only about 0.005 at $(10^3, 50)$. For $M = 10^3$, this gap remains around 0.005 to 0.006 across all horizons, so TS-PS-1 is already very close to the oracle policy.

The left tails of the distributions tell a similar story. UCB-PS is the most left-shifted algorithm in both panels and has the poorest lower-tail behavior. Myopic-PS also exhibits a noticeable lower tail, reflecting its tendency to become trapped in suboptimal allocations. TS-PS-1 still has some lower-tail risk because its prior is learned from finite data, but this risk is much smaller when $M = 10^3$. For example, at $N = 50$ the 10th percentile of the PR under TS-PS-1 increases from 0.850 when $M = 10$ to 0.920 when $M = 10^3$. Overall, the enlarged experiment yields a clear message. The qualitative ranking of the algorithms is unchanged, but a larger batch size greatly improves the value of prior learning. TS-PS-∞ remains best, TS-PS-1 becomes nearly as good as the oracle policy in the large-batch regime, TS-PS-0 is a strong cold-start baseline, myopic-PS is competitive only at short horizons, and UCB-PS performs worst in these heterogeneous random instances.

6 Conclusion

We studied a finite-horizon online resource allocation problem for service platforms that must allocate tasks under hard server capacities while learning uncertain matching rewards. A central message of the paper is that an LP benchmark is useful only if its recommendation can be executed online in a pathwise-feasible way. In the homogeneous model, this leads to a feasibility-aware execution layer together with two complementary learning policies: UCB with cumulative tracking (UCB-CT), which achieves logarithmic instance-dependent regret together with an $O(\sqrt{JN \log N})$ worst-case regret bound, equivalently $\widetilde{O}(\sqrt{JN})$, and Thompson sampling with probability sampling (TS-PS), which achieves $O(\sqrt{N})$ Bayesian regret using only realized-assignment feedback.

We then extended the framework to a heterogeneous setting in which a fixed panel of M customers contributes one task in each of N assignment rounds and the customers belong to latent types. This setting requires learning at the customer level, since observations from early assignments should inform later assignments for the same customer. To capture this structure, we introduced a customer-level benchmark and posterior model, showed that the customer-level LP benchmark approaches the type-level LP benchmark as the batch size grows, and proposed TS-PS-Heterogeneous with a single-batch Bayesian regret bound of order $\sqrt{N} + (J-1)\sqrt{JN/M}$ under realized-assignment feedback. To support repeated deployment across batches, we further developed a clustering–modeling–assignment pipeline that estimates latent reward parameters and type proportions from historical service data, converts them into informative priors, and reuses them in subsequent batches. Under identifiability and sufficient exploration, the accelerated EM procedure recovers the realized clustering in two rounds with high probability and yields consistent estimates of the latent population parameters.

Our numerical study further clarifies the algorithmic lessons. First, execution matters: cumulative

tracking and probability sampling pace scarce capacity more effectively than direct tracking, which can overreact to unstable LP decompositions and deplete high-value servers too early. Second, latent structure matters: informative latent-type priors substantially improve performance across batches and rapidly approach oracle benchmarks, whereas pooling the same learned information into a non-latent warm start can underperform even a cold start. Across the tested instances, Thompson-sampling-based policies were generally the most robust, especially when combined with informative priors.

Taken together, these results suggest that online resource allocation for service platforms should be viewed as a joint estimation–optimization–execution problem rather than as reward learning alone. Relative to generic constrained-bandit formulations, the structure studied here yields sharper insights: deterministic identity consumption in the homogeneous model enables pathwise-feasible LP tracking, while latent customer structure in the heterogeneous model creates a natural mechanism for within-customer learning and across-batch prior transfer. Several directions remain open. On the theory side, it would be valuable to obtain stronger problem-dependent and frequentist guarantees in the heterogeneous setting, especially for UCB-style policies, and to develop end-to-end repeated-batch guarantees for the clustering–modeling–assignment pipeline under estimation error, model misspecification, and evolving customer populations. On the modeling side, extensions to queueing dynamics, time-varying capacities, and richer covariates would broaden the scope of the framework. We hope the tools developed here provide a useful foundation for these directions.

Acknowledgments

Wei You's research is generously supported by the Hong Kong Research Grants Council [Grant GRF 16212823] and [Grant ECS 26212320].

References

Rajeev Agrawal, Demosthenis Teneketzis, and Venkatachalam Anantharam. Asymptotically efficient adaptive allocation schemes for controlled markov chains: Finite parameter space. *IEEE Transactions on Automatic Control*, 34(12):1249–1259, 1989.

Shipra Agrawal and Nikhil R. Devanur. Linear contextual bandits with global constraints and objective. *arXiv preprint arXiv:1507.06738*, 2015.

Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

Shipra Agrawal, Nikhil R. Devanur, and Lihong Li. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In *Proceedings of the 29th Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 4–18. PMLR, PMLR, 2016.

Venkatachalam Anantharam, Pravin Varaiya, and Jean Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i: Iid rewards. *IEEE Transactions on Automatic Control*, 32(11):968–976, 1987.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *Journal of the ACM*, 65(3), 2018. doi: 10.1145/3164539.

Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations research*, 57(6):1407–1420, 2009.

Omar Besbes and Assaf Zeevi. Blind network revenue management. *Operations research*, 60(6): 1537–1550, 2012.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

Yiwei Chen and Cong Shi. Network revenue management with online inverse batch gradient descent method. *Production and Operations Management*, 2023.

Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, et al. Combinatorial bandits revisited. *Advances in neural information processing systems*, 28, 2015.

Sanjoy Dasgupta and Leonard J Schulman. A probabilistic analysis of em for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8:203–226, 2007.

Rémy Degenne, Wouter M Koolen, and Pierre Ménard. Non-asymptotic pure exploration by solving games. *Advances in Neural Information Processing Systems*, 32, 2019.

Rémy Degenne, Han Shao, and Wouter Koolen. Structure adaptive algorithms for stochastic bandits. In *International Conference on Machine Learning*, pages 2443–2452. PMLR, 2020.

Arnoud V Den Boer. Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in operations research and management science*, 20(1):1–18, 2015.

Kris Johnson Ferreira, David Simchi-Levi, and He Wang. Online network revenue management using thompson sampling. *Operations Research*, 66(6):1586–1602, 2018. doi: 10.1287/opre.2018.1755.

Arthur Flajolet and Patrick Jaillet. Logarithmic regret bounds for bandits with knapsacks. *arXiv preprint arXiv:1510.01800*, 2015.

Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027. PMLR, 2016.

Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *International conference on machine learning*, pages 757–765. PMLR, 2014.

Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. On context-dependent clustering of bandits. In *International Conference on machine learning*, pages 1253–1262. PMLR, 2017.

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

Wei-Kang Hsu, Jiaming Xu, Xiaojun Lin, and Mark R Bell. Integrated online learning and adaptive control in queueing systems with uncertain payoffs. *Operations Research*, 70(2):1166–1181, 2022.

Ramesh Johari, Vijay Kamble, and Yash Kanoria. Matching while learning. *Operations Research*, 69(2):655–681, 2021.

Anand Kalvit and Assaf Zeevi. Dynamic learning in large matching markets. *Advances in Neural Information Processing Systems*, 35:20770–20781, 2022.

Jung-hun Kim and Milan Vojnovic. Scheduling servers with stochastic bilinear rewards. *arXiv preprint arXiv:2112.06362*, 2021.

Junpei Komiyama, Junya Honda, and Hiroshi Nakagawa. Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *International Conference on Machine Learning*, pages 1152–1161. PMLR, 2015.

Tze Leung Lai, Herbert Robbins, et al. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Xiaocheng Li, Chunlin Sun, and Yinyu Ye. The symmetry between arms and knapsacks: A primal-dual approach for bandits with knapsacks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6483–6492. PMLR, PMLR, 2021.

Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In *International Conference on Machine Learning*, pages 136–144. PMLR, 2014.

Laurent Massoulié and Kuang Xu. On the capacity of information processing systems. In *Conference on Learning Theory*, pages 1292–1297. PMLR, 2016.

Sentao Miao, Xi Chen, Xiuli Chao, Jiaxi Liu, and Yidong Zhang. Context-based dynamic pricing with online clustering. *Production and Operations Management*, 31(9):3559–3575, 2022.

Aldo Pacchiano, Mohammad Ghavamzadeh, Peter Bartlett, and Heinrich Jiang. Stochastic bandits with linear constraints. In *International conference on artificial intelligence and statistics*, pages 2827–2835. PMLR, 2021.

Soumyabrata Pal, Arun Sai Suggala, Karthikeyan Shanmugam, and Prateek Jain. Optimal algorithms for latent bandits with cluster structure. In *International Conference on Artificial Intelligence and Statistics*, pages 7540–7577. PMLR, 2023.

Karthik Abinav Sankararaman and Aleksandrs Slivkins. Bandits with knapsacks beyond the worst case. In *Advances in Neural Information Processing Systems 34*, pages 23191–23204. NeurIPS, 2021.

Vidit Saxena, Joakim Jalden, and Joseph Gonzalez. Thompson sampling for linearly constrained bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 1999–2009. PMLR, 2020.

Xu Sun, Shiwei Chai, and Jingtong Zhao. Congestion-aware matching and learning for service platforms. *Available at SSRN 5258944*, 2023.

Huasen Wu, Rayadurgam Srikant, Xin Liu, and Chong Jiang. Algorithms with logarithmic or sublinear regret for constrained contextual bandits. *Advances in Neural Information Processing Systems*, 28, 2015.

Appendix

A UCB with Direct Tracking

This appendix discusses the direct-tracking UCB variant for the homogeneous model. Relative to UCB-CT, UCB-DT keeps the same optimistic estimation layer but executes the empirical LP through direct tracking rather than cumulative tracking. The resulting analysis is sharper in separated instances and yields logarithmic regret, whereas the cleaner gap-free argument is carried by UCB-CT in the main text. Throughout this section, we assume $J \geq 2$, $N \geq J$, and $r_1 > r_2 > \cdots > r_J$. When $N = J$, only the initialization phase is executed; the UCB-DT policy and its analysis are not asserted for $N < J$. For each round $n > J$ and server $j \in [J]$, define the empirical mean and the UCB index

$$\widehat{r}_j^n = \frac{\sum_{s=1}^{n-1} X_{j_s}^s \mathbb{1}(j_s = j)}{N_j^{n-1}} = \frac{\sum_{s<n:j_s=j} X_{j_s}^s}{N_j^{n-1}} \qquad \text{and} \qquad C_j^n = \widehat{r}_j^n + \sqrt{\frac{2\log(n-1)}{N_j^{n-1}}}.$$

Because each server is sampled once during initialization, $N_j^{n-1} \geq 1$ for all $j \in [J]$ and $n > J$.

The next comparison lemma is the basic concentration input for UCB-DT.

Lemma 6. *For any $n \geq J$ and any $j, j' \in [J]$ with $r_j < r_{j'}$,*

$$\mathbb{P}\left(C_j^{n+1} \geq C_{j'}^{n+1}, N_j^n \geq \frac{8\log n}{(r_{j'} - r_j)^2}\right) \leq \frac{2}{n^3}.$$

Lemma 6 says that once a lower-reward server has been sampled $O(\log N)$ times, its UCB index is unlikely to exceed that of a strictly better server. A full proof is given in Appendix A.1.

We now combine UCB with direct tracking. At each round $n > J$, the platform computes the UCB indices $\{C_j^n\}_{j=1}^J$, solves the LP obtained by replacing r_j with C_j^n, and then direct-tracks the resulting allocation vector. We refer to the resulting policy as UCB with direct tracking, or UCB-DT.

Lemma 7. *Under UCB-DT, the selected server always satisfies $B_{j_n}^n > 0$ for all $n \in [N]$ so that the policy is admissible. Moreover, for every $n = J+1, \ldots, N$, we have $p_{j_n}^n > 0$.*

Theorem 5. *Assume $r_1 > r_2 > \cdots > r_J$. There exist constants $C_1(\boldsymbol{r}, \boldsymbol{\mu}), C_2(\boldsymbol{r}, \boldsymbol{\mu}) > 0$ such that*

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq C_1(\boldsymbol{r}, \boldsymbol{\mu}) \log N + C_2(\boldsymbol{r}, \boldsymbol{\mu}).$$

In particular, UCB-DT achieves $O(\log N)$ regret.

The proof follows the same structural template as the gap-dependent UCB-CT analysis. First, every zero-mass server is shown to be selected only $O(\log N)$ times in expectation. Second, when $p_{\widetilde{j}+1}^* > 0$, the critical server $\widetilde{j}+1$ can exceed its LP target by at most a constant in expectation. A deficit identity then transfers these allocation bounds to the saturated servers and yields logarithmic regret. The full proof is given in Appendix A.1.

Algorithm 6 UCB with direct tracking (UCB-DT)

for $n = 1, \ldots, J$ **do**
  Select server $j_n = n$ and observe $X^n_{j_n}$.
end for
for $n = J+1, \ldots, N$ **do**
  Compute
$$C^n_j = \frac{\sum_{s=1}^{n-1} X^s_{j_s} \mathbb{1}(j_s = j)}{N^{n-1}_j} + \sqrt{\frac{2\log(n-1)}{N^{n-1}_j}}, \qquad j \in [J].$$
  Compute $\boldsymbol{p}^n \in \operatorname{argmax}_{\boldsymbol{p}\in\mathbb{R}^J} \left\{ \sum_{j=1}^J p_j C^n_j : 0 \le p_j \le \mu_j \text{ for all } j, \sum_{j=1}^J p_j = 1 \right\}$.
  Select $j_n \in \operatorname{argmax}_{j\in[J]} \left\{ n p^n_j - N^{n-1}_j \right\}$.
  Observe $X^n_{j_n}$.
end for

A.1 Proofs for UCB-DT

This appendix proves the claims stated in Section A. We reuse Lemmas 10–11 from the common preliminaries in Appendix C; once the UCB indices are controlled, these generic exchange lemmas provide all LP-structure arguments needed for DT.

A.1.1 Basic properties

Proof of Lemma 6. Fix $j, j' \in [J]$ with $r_j < r_{j'}$, and write $\Delta = r_{j'} - r_j > 0$. Define $E_n = \left\{ C^{n+1}_j \ge C^{n+1}_{j'}, N^n_j \ge \frac{8\log n}{\Delta^2} \right\}$. If E_n occurs and both

$$\overline{Y}_{j,N^n_j} \le r_j + \sqrt{\frac{2\log n}{N^n_j}} \quad \text{and} \quad \overline{Y}_{j',N^n_{j'}} \ge r_{j'} - \sqrt{\frac{2\log n}{N^n_{j'}}}$$

hold, then

$$C^{n+1}_j \le r_j + 2\sqrt{\frac{2\log n}{N^n_j}} \le r_j + \Delta = r_{j'},$$

where the second inequality uses $N^n_j \ge 8\log n/\Delta^2$. At the same time, we have $C^{n+1}_{j'} \ge r_{j'}$. Hence $C^{n+1}_j \le C^{n+1}_{j'}$, a contradiction. Therefore E_n implies that at least one of the following two deviations occurs:

$$\overline{Y}_{j,N^n_j} > r_j + \sqrt{\frac{2\log n}{N^n_j}}, \qquad \overline{Y}_{j',N^n_{j'}} < r_{j'} - \sqrt{\frac{2\log n}{N^n_{j'}}}.$$

By Hoeffding's inequality and a union bound over the possible sample counts,

$$\mathbb{P}\left(\overline{Y}_{j,N^n_j} > r_j + \sqrt{\frac{2\log n}{N^n_j}} \right) \le \sum_{s=1}^n \mathbb{P}\left(\overline{Y}_{j,s} > r_j + \sqrt{\frac{2\log n}{s}} \right) \le \frac{1}{n^3},$$

and similarly,

$$\mathbb{P}\left(\overline{Y}_{j',N^n_{j'}} < r_{j'} - \sqrt{\frac{2\log n}{N^n_{j'}}} \right) \le \frac{1}{n^3}.$$

Thus $\mathbb{P}(E_n) \le 2/n^3$, which proves the lemma. □

Proof of Lemma 7. For $n \le J$, feasibility is immediate from the initialization phase because each server has initial capacity at least one.

Fix now any round $n > J$. Since $\sum_{j=1}^{J} p_j^n = 1$ and $\sum_{j=1}^{J} N_j^{n-1} = n-1$, we have $\sum_{j=1}^{J} \left(np_j^n - N_j^{n-1}\right) = 1$. If $B_j^n = 0$, then $N_j^{n-1} = c_j$, so $np_j^n - N_j^{n-1} = np_j^n - c_j \le 0$, because $n \le N$ and $p_j^n \le \mu_j$. Hence every unavailable server has nonpositive discrepancy, whereas the discrepancies sum to 1. Therefore any maximizer of $np_j^n - N_j^{n-1}$ must be available, proving $B_{j_n}^n > 0$.

If $p_j^n = 0$, then $np_j^n - N_j^{n-1} = -N_j^{n-1} \le 0$. Again, because the discrepancies sum to 1, their maximum is strictly positive. Hence any maximizer must satisfy $p_j^n > 0$. This proves the lemma. □

A.1.2 Allocation bounds

The first step mirrors the gap-dependent UCB-CT analysis: once a zero-mass server has been sampled past its logarithmic threshold, the empirical LP assigns it zero mass and direct tracking can no longer select it.

Proposition 5. *Let $\kappa_0 = 2\sum_{n=1}^{\infty} \frac{1}{n^3} < \infty$. If $p^*_{\widetilde{j}+1} > 0$, then for every $j \ge \widetilde{j} + 2$,*

$$\mathbb{E}[N_j^N] \le 1 + \left\lceil \frac{8 \log N}{(r_{\widetilde{j}+1} - r_j)^2} \right\rceil + \kappa_0(\widetilde{j} + 1).$$

If $p^*_{\widetilde{j}+1} = 0$, then for every $j \ge \widetilde{j} + 1$,

$$\mathbb{E}[N_j^N] \le 1 + \left\lceil \frac{8 \log N}{(r_{\widetilde{j}} - r_j)^2} \right\rceil + \kappa_0 \widetilde{j}.$$

In particular, $\mathbb{E}[N_j^N] = O(\log N)$ for every server j such that $p_j^* = 0$.

Proof of Proposition 5. Fix a zero-mass server j and define

$$m_j = \begin{cases} \widetilde{j} + 1, & p^*_{\widetilde{j}+1} > 0, \\ \widetilde{j}, & p^*_{\widetilde{j}+1} = 0, \end{cases} \qquad \Delta_j = r_{m_j} - r_j > 0, \qquad u_j(n) = \frac{8 \log n}{\Delta_j^2}.$$

Then $r_i > r_j$ for every $i = 1, \dots, m_j$ and $\sum_{i=1}^{m_j} \mu_i \ge 1$. Let

$$G_n(j) = \left\{ C_i^{n+1} > C_j^{n+1} \text{ for all } i = 1, \dots, m_j \right\}.$$

On $G_n(j)$, Lemma 10 implies that every optimal empirical LP solution satisfies $p_j^{n+1} = 0$. By Lemma 7, this forces $j_{n+1} \ne j$. Hence

$$\left\{ j_{n+1} = j,\ N_j^n \ge u_j(n) \right\} \subseteq G_n(j)^c.$$

For each $i = 1, \dots, m_j$, since $r_i - r_j \ge \Delta_j$, Lemma 6 yields

$$\mathbb{P}\left(C_j^{n+1} \ge C_i^{n+1},\ N_j^n \ge u_j(n) \right) \le \frac{2}{n^3}.$$

A union bound therefore gives

$$\mathbb{P}\left(G_n(j)^c,\ N_j^n \geq u_j(n)\right) \leq \frac{2m_j}{n^3}.$$

Since $u_j(n)$ is increasing in n, pathwise

$$N_j^N \leq 1 + \lceil u_j(N) \rceil + \sum_{n=J}^{N-1} \mathbb{1}\left(j_{n+1} = j,\ N_j^n \geq u_j(n)\right).$$

Taking expectations gives

$$\mathbb{E}[N_j^N] \leq 1 + \left\lceil \frac{8\log N}{\Delta_j^2} \right\rceil + 2m_j \sum_{n=J}^{\infty} \frac{1}{n^3} \leq 1 + \left\lceil \frac{8\log N}{\Delta_j^2} \right\rceil + \kappa_0 m_j.$$

Unpacking m_j and Δ_j in the two cases proves the stated bounds. □

When $p^*_{\widetilde{j}+1} > 0$, the server $\widetilde{j}+1$ is the unique server with strictly positive but nonsaturated LP mass. The next lemma shows that its overshoot above the LP target is only $O(1)$ in expectation.

Lemma 8. *Assume $\widetilde{j} \geq 1$ and $p^*_{\widetilde{j}+1} > 0$. Let $\delta = p^*_{\widetilde{j}+1} = 1 - S_{\widetilde{j}}$ and $\Delta_c = r_{\widetilde{j}} - r_{\widetilde{j}+1}$. Then there exists a constant $K(\boldsymbol{r}, \boldsymbol{\mu}) \in \mathbb{N}$ such that*

$$\mathbb{E}[N^N_{\widetilde{j}+1}] \leq \delta N + K(\boldsymbol{r}, \boldsymbol{\mu}) + 1 + \kappa_0 \widetilde{j}.$$

Proof of Lemma 8. Let $c = \widetilde{j}+1$. Because $\log n/n \to 0$, there exists $K(\boldsymbol{r}, \boldsymbol{\mu})$ such that

$$\delta(n+1) \geq \frac{8\log n}{\Delta_c^2} \qquad \text{for all } n \geq K(\boldsymbol{r}, \boldsymbol{\mu}). \tag{7}$$

For $n \geq K(\boldsymbol{r}, \boldsymbol{\mu})$, define

$$H_n = \left\{C_i^{n+1} > C_c^{n+1} \text{ for all } i = 1, \ldots, \widetilde{j}\right\}.$$

On H_n, Lemma 11 implies that every optimal empirical LP solution satisfies $p_c^{n+1} \leq \delta$. Therefore, on $H_n \cap \{N_c^n \geq (n+1)\delta\}$,

$$(n+1)p_c^{n+1} - N_c^n \leq 0.$$

Because the direct-tracking discrepancies sum to 1, server c cannot be selected on this event. Hence

$$\{j_{n+1} = c,\ N_c^n \geq (n+1)\delta\} \subseteq H_n^c.$$

By (7), the event $\{N_c^n \geq (n+1)\delta\}$ implies $N_c^n \geq 8\log n/\Delta_c^2$. Applying Lemma 6 and a union bound over $i = 1, \ldots, \widetilde{j}$ yields

$$\mathbb{P}\left(H_n^c,\ N_c^n \geq (n+1)\delta\right) \leq \frac{2\widetilde{j}}{n^3}, \qquad n \geq K(\boldsymbol{r}, \boldsymbol{\mu}).$$

It remains to bound the number of selections while $N_c^n < (n+1)\delta$. Define

$$M = \sum_{n=0}^{N-1} \mathbb{1}\left(j_{n+1} = c,\ N_c^n < (n+1)\delta\right).$$

If the q-th such selection occurs at time $n_q + 1$, then before that pull we have

$$q - 1 \le N_c^{n_q} < (n_q + 1)\delta \le N\delta,$$

so $q \le \lfloor N\delta \rfloor + 1$. Thus $M \le \lfloor N\delta \rfloor + 1$ pathwise. Now decompose

$$N_c^N = M + \sum_{n=0}^{N-1} \mathbb{1}\left(j_{n+1} = c,\ N_c^n \ge (n+1)\delta\right).$$

The second sum has at most $K(\boldsymbol{r}, \boldsymbol{\mu})$ terms with $n < K(\boldsymbol{r}, \boldsymbol{\mu})$. Taking expectations and using the previous probability bound gives

$$\mathbb{E}[N_c^N] \le \delta N + 1 + K(\boldsymbol{r}, \boldsymbol{\mu}) + 2\widetilde{j} \sum_{n=K(\boldsymbol{r},\boldsymbol{\mu})}^{\infty} \frac{1}{n^3} \le \delta N + 1 + K(\boldsymbol{r}, \boldsymbol{\mu}) + \kappa_0 \widetilde{j}.$$

This proves the lemma. □

A.1.3 Proof of Theorem 5

Proof of Theorem 5. We distinguish three cases.

Case 1: $\widetilde{j} = 0$. In this case, $p_1^* = 1$ and $p_j^* = 0$ for all $j \ge 2$, so

$$R^* = Nr_1 \qquad \text{and} \qquad R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] = \sum_{j=2}^{J} (r_1 - r_j)\mathbb{E}[N_j^N].$$

Applying Proposition 5 yields $\mathbb{E}[N_j^N] = O(\log N)$ for each $j \ge 2$, hence the regret is $O(\log N)$.

Case 2: $\widetilde{j} \ge 1$ and $p_{\widetilde{j}+1}^* > 0$. Let

$$c = \widetilde{j} + 1, \qquad \delta = p_c^* = 1 - S_{\widetilde{j}}, \qquad Z = \{c+1, \dots, J\}.$$

For $i \le \widetilde{j}$, define the saturated-server deficit

$$D_i = c_i - \mathbb{E}[N_i^N] \ge 0.$$

Because

$$\sum_{i=1}^{\widetilde{j}} c_i + \delta N = N \quad \text{and} \quad \sum_{i=1}^{\widetilde{j}} \mathbb{E}[N_i^N] + \mathbb{E}[N_c^N] + \sum_{j \in Z} \mathbb{E}[N_j^N] = N,$$

we obtain the identity

$$\sum_{i=1}^{\widetilde{j}} D_i = \mathbb{E}[N_c^N] - \delta N + \sum_{j \in Z} \mathbb{E}[N_j^N]. \tag{8}$$

Now expand the regret:

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] = \sum_{i=1}^{\widetilde{j}} (r_i - r_c) D_i + \sum_{j \in Z} (r_c - r_j)\mathbb{E}[N_j^N].$$

Using $D_i \geq 0$ and (8),

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq (r_1 - r_c)\left(\mathbb{E}[N_c^N] - \delta N + \sum_{j\in Z}\mathbb{E}[N_j^N]\right) + \sum_{j\in Z}(r_c - r_j)\mathbb{E}[N_j^N].$$

By Lemma 8, $\mathbb{E}[N_c^N] - \delta N = O(1)$. By Proposition 5, $\mathbb{E}[N_j^N] = O(\log N)$ for every $j \in Z$. Hence the regret is $O(\log N)$.

Case 3: $p^*_{\widetilde{j}+1} = 0$. Now $S_{\widetilde{j}} = 1$ and every server in $Z = \{\widetilde{j}+1, \ldots, J\}$ is zero-mass. Define again $D_i = c_i - \mathbb{E}[N_i^N] \geq 0$ for $i = 1, \ldots, \widetilde{j}$. Since

$$\sum_{i=1}^{\widetilde{j}} c_i = N \quad \text{and} \quad \sum_{i=1}^{\widetilde{j}}\mathbb{E}[N_i^N] + \sum_{j\in Z}\mathbb{E}[N_j^N] = N,$$

we have

$$\sum_{i=1}^{\widetilde{j}} D_i = \sum_{j\in Z}\mathbb{E}[N_j^N]. \tag{9}$$

Therefore

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] = \sum_{i=1}^{\widetilde{j}}(r_i - r_{\widetilde{j}})D_i + \sum_{j\in Z}(r_{\widetilde{j}} - r_j)\mathbb{E}[N_j^N] \leq (r_1 - r_{\widetilde{j}})\sum_{i=1}^{\widetilde{j}} D_i + \sum_{j\in Z}(r_{\widetilde{j}} - r_j)\mathbb{E}[N_j^N].$$

Using (9) and Proposition 5, we conclude that the regret is $O(\log N)$.

Combining the three cases proves the theorem. □

B Proofs for The LP and Oracle Tracking

Lemma 9 (Theorem 6 of Degenne et al. (2020))**.** *Let $\{\boldsymbol{p}^n\}_{n\geq 1}$ be any sequence of probability vectors in $\mathbb{R}^J$. Define $\boldsymbol{P}^0 = \boldsymbol{N}^0 = \mathbf{0}$ and*

$$P_j^n = \sum_{s=1}^{n} p_j^s, \qquad N_j^n = \sum_{s=1}^{n} \mathbb{1}(j_s = j), \qquad j \in [J], \quad n \geq 1.$$

If $j_n \in \operatorname{argmax}_{j\in[J]}\left\{P_j^n - N_j^{n-1}\right\}$, then, for every $j \in [J]$ and every $n \geq 1$,

$$-\sum_{i=2}^{J}\frac{1}{i} \leq N_j^n - P_j^n \leq 1.$$

Proof of Lemma 1. Let $\boldsymbol{p}$ be any optimal solution to (1). Because

$$\sum_{i=1}^{\widetilde{j}+1}\mu_i > 1 \quad \text{and} \quad r_i > r_j \text{ for all } i \leq \widetilde{j}+1 < j,$$

Lemma 10 implies that $p_j = 0$ for all $j \geq \widetilde{j}+2$. If $\widetilde{j} = 0$, then the preceding display already gives $p_j = 0$ for all $j \geq 2$, so $p_1 = 1$ by feasibility. This is exactly (2). If $\widetilde{j} \geq 1$, apply Lemma 11 with

upper bounds $u_i = \mu_i$ and coefficients $a_i = r_i$. Because $r_i > r_{\widetilde{j}+1}$ for all $i \le \widetilde{j}$, every optimal solution satisfies $p_{\widetilde{j}+1} \le 1 - S_{\widetilde{j}}$. Since the remaining coordinates vanish and $\sum_{j=1}^{J} p_j = 1$, we must have

$$\sum_{i=1}^{\widetilde{j}} p_i = 1 - p_{\widetilde{j}+1} \ge S_{\widetilde{j}} = \sum_{i=1}^{\widetilde{j}} \mu_i.$$

Because also $p_i \le \mu_i$ for each $i \le \widetilde{j}$, it follows that $p_i = \mu_i$ for $i = 1, \dots, \widetilde{j}$, and therefore $p_{\widetilde{j}+1} = 1 - S_{\widetilde{j}}$. This proves (2). Uniqueness follows from the strict ordering $r_1 > \cdots > r_J$. □

Proof of Lemma 2. With deterministic tie-breaking, j_n is measurable with respect to $\mathcal{F}_{n-1}$, because it depends only on $\boldsymbol{p}^*$ and the past allocation counts $\{N_j^{n-1}\}_{j=1}^{J}$. It remains to show that the selected server is always available.

Define $P_j^n = np_j^*$ for each $j \in [J]$ and $n \in [N]$. Suppose, for contradiction, that $B_{j_n}^n = 0$ for some round n. Then $N_{j_n}^{n-1} = c_{j_n}$. Since $p_{j_n}^* \le \mu_{j_n}$ and $n \le N$, we have $P_{j_n}^n - N_{j_n}^{n-1} = np_{j_n}^* - c_{j_n} \le 0$. Because j_n maximizes $P_j^n - N_j^{n-1}$, it follows that $P_j^n - N_j^{n-1} \le 0$ for every $j \in [J]$. However,

$$\sum_{j=1}^{J} \left(P_j^n - N_j^{n-1} \right) = \sum_{j=1}^{J} np_j^* - \sum_{j=1}^{J} N_j^{n-1} = n - (n-1) = 1,$$

which is impossible. Therefore $B_{j_n}^n > 0$ for all $n \in [N]$. □

Proof of Proposition 2. Let $H_J = \sum_{i=2}^{J} \frac{1}{i}$. Apply Lemma 9 with $\boldsymbol{p}^n \equiv \boldsymbol{p}^*$ for all $n \in [N]$. Then, for every $j \in [J]$, $N_j^N \ge Np_j^* - H_J$. Using the definition of R^* and the identity $\mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] = \sum_{j=1}^{J} r_j \mathbb{E}[N_j^N]$, we obtain

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] = N \sum_{j=1}^{J} p_j^* r_j - \sum_{j=1}^{J} r_j \mathbb{E}[N_j^N] = \sum_{j=1}^{J} r_j \left(Np_j^* - \mathbb{E}[N_j^N] \right) \le H_J \sum_{j=1}^{J} r_j.$$

This proves the claim. □

C Proofs for UCB-CT

We use the same residual-phase notation for both the gap-dependent and worst-case analyses. Let $T = N - J$. For each residual round $t \in [T]$, write $n = J + t$ and define

$$M_j^t = N_j^{J+t} - 1, \quad M_j^0 = 0, \qquad \text{and} \qquad Q_j^t = \sum_{s=J+1}^{J+t} q_j^s, \quad Q_j^0 = 0.$$

Thus M_j^t is the number of residual tasks assigned to server j by the end of residual round t, and Q_j^t is its cumulative residual target. We also write

$$R_{\text{res}}^* = T \max_{\boldsymbol{q} \in \mathbb{R}^J} \left\{ \sum_{j=1}^{J} q_j r_j : 0 \le q_j \le \bar{\mu}_j \text{ for all } j, \sum_{j=1}^{J} q_j = 1 \right\}.$$

Finally, let

$$\Gamma_T = R_{\text{res}}^* - \sum_{j=1}^{J} r_j Q_j^T.$$

The quantity Γ_T is the residual target-selection loss.

C.1 Common preliminaries

The next two exchange lemmas describe the structure of the empirical LP.

Lemma 10. *Consider the LP*

$$\max_{\boldsymbol{p}\in\mathbb{R}^J}\left\{\sum_{i=1}^J p_i a_i : 0\le p_i\le u_i \text{ for all } i, \sum_{i=1}^J p_i = 1\right\},$$

where $\boldsymbol{a}\in\mathbb{R}^J$ and $\boldsymbol{u}\in\mathbb{R}^J_+$ are arbitrary. Fix $j\in[J]$. If there exists a set $S\subseteq[J]\setminus\{j\}$ such that

$$a_i > a_j \quad \text{for all } i\in S, \qquad \sum_{i\in S} u_i \ge 1,$$

then every optimal solution satisfies $p_j = 0$.

Proof of Lemma 10. Suppose, for contradiction, that some optimal solution $\boldsymbol{p}$ satisfies $p_j > 0$. Since $\sum_{i\in S} u_i \ge 1$ and $\sum_{i=1}^J p_i = 1$, the inequality $p_j > 0$ implies

$$\sum_{i\in S} p_i < 1 \le \sum_{i\in S} u_i.$$

Hence there exists $i^\star\in S$ such that $p_{i^\star} < u_{i^\star}$. Let $\varepsilon = \min\{p_j, u_{i^\star} - p_{i^\star}\} > 0$. Define $\widetilde{\boldsymbol{p}}$ by moving mass ε from j to $i^\star$:

$$\widetilde{p}_{i^\star} = p_{i^\star} + \varepsilon, \qquad \widetilde{p}_j = p_j - \varepsilon, \qquad \widetilde{p}_i = p_i \text{ for } i\notin\{i^\star, j\}.$$

Then $\widetilde{\boldsymbol{p}}$ is feasible and

$$\sum_{i=1}^J \widetilde{p}_i a_i - \sum_{i=1}^J p_i a_i = \varepsilon(a_{i^\star} - a_j) > 0,$$

contradicting optimality of $\boldsymbol{p}$. Therefore every optimal solution must satisfy $p_j = 0$. □

Lemma 11. *Let*

$$k = \max\left\{m\in\{0,\dots,J\} : \sum_{i=1}^m u_i \le 1\right\}, \qquad \delta_u = 1 - \sum_{i=1}^k u_i.$$

Assume $k\ge 1$. Consider the LP

$$\max_{\boldsymbol{p}\in\mathbb{R}^J}\left\{\sum_{i=1}^J p_i a_i : 0\le p_i\le u_i \text{ for all } i, \sum_{i=1}^J p_i = 1\right\}.$$

If $a_i > a_{k+1}$ for all $i = 1,\dots,k$, then every optimal solution satisfies $p_{k+1}\le\delta_u$.

Proof of Lemma 11. Suppose, for contradiction, that some optimal solution $\boldsymbol{p}$ satisfies $p_{k+1} > \delta_u$. Then

$$\sum_{i=1}^k p_i = 1 - p_{k+1} - \sum_{i=k+2}^J p_i < 1 - \delta_u = \sum_{i=1}^k u_i.$$

Hence there exists $i^\star\in\{1,\dots,k\}$ such that $p_{i^\star} < u_{i^\star}$. Let $\varepsilon = \min\{p_{k+1} - \delta_u, u_{i^\star} - p_{i^\star}\} > 0$. Move mass ε from $k+1$ to $i^\star$. The resulting vector is feasible and strictly improves the objective because $a_{i^\star} > a_{k+1}$. This contradicts optimality. Therefore every optimal solution must satisfy $p_{k+1}\le\delta_u$. □

Lemma 12. *The original LP benchmark and the LP benchmark satisfy* $R^* \le J + R^*_{\mathrm{res}}$.

Proof of Lemma 12. Let $\boldsymbol{p}^*$ be an optimal solution to (1), and define

$$x_j^* = Np_j^*, \qquad j \in [J].$$

Then

$$0 \le x_j^* \le c_j, \qquad \sum_{j=1}^{J} x_j^* = N, \qquad R^* = \sum_{j=1}^{J} x_j^* r_j.$$

Define

$$y_j = (x_j^* - 1)_+, \qquad j \in [J].$$

Since $(a-1)_+ \ge a - 1$ for all $a \ge 0$,

$$\sum_{j=1}^{J} y_j \ge \sum_{j=1}^{J} (x_j^* - 1) = N - J = T.$$

Also, since $x_j^* \le c_j$,

$$y_j \le c_j - 1 = b_j.$$

Hence we can choose numbers $z_j \in [0, y_j]$ such that

$$\sum_{j=1}^{J} z_j = T.$$

Then $\boldsymbol{z}/T$ is feasible for the LP, so

$$R^*_{\mathrm{res}} \ge \sum_{j=1}^{J} z_j r_j.$$

Finally,

$$\sum_{j=1}^{J} (x_j^* - z_j) = N - T = J.$$

Since $0 \le r_j \le 1$ for all j,

$$\sum_{j=1}^{J} z_j r_j \ge \sum_{j=1}^{J} x_j^* r_j - J = R^* - J.$$

Therefore

$$R^*_{\mathrm{res}} \ge R^* - J,$$

which proves the lemma. □

For each server $j \in [J]$, let $\{Y_{j,s}\}_{s \ge 1}$ denote the sequence of rewards generated by the successive pulls of server j, and define

$$\overline{Y}_{j,s} = \frac{1}{s} \sum_{\ell=1}^{s} Y_{j,\ell}.$$

Lemma 13. *Define the event*

$$\mathcal{E} = \left\{ \left| \overline{Y}_{j,s} - r_j \right| \le \sqrt{\frac{2\log N}{s}} \text{ for all } j \in [J],\ s \in [N] \right\}.$$

Then

$$\mathbb{P}(\mathcal{E}^c) \le \frac{2J}{N^3}.$$

Moreover, on $\mathcal{E}$, for every $j \in [J]$ and every $t \in [T]$,

$$r_j \le C_j^{J+t} \le r_j + 2\sqrt{\frac{2\log N}{1+M_j^{t-1}}}.$$

Proof of Lemma 13. For any fixed $j \in [J]$ and $s \in [N]$, Hoeffding's inequality gives

$$\mathbb{P}\left(\left| \overline{Y}_{j,s} - r_j \right| > \sqrt{\frac{2\log N}{s}} \right) \le 2\exp\left(-2s \cdot \frac{2\log N}{s} \right) = \frac{2}{N^4}.$$

A union bound over all $j \in [J]$ and $s \in [N]$ yields

$$\mathbb{P}(\mathcal{E}^c) \le JN \cdot \frac{2}{N^4} = \frac{2J}{N^3}.$$

At the beginning of residual round t, server j has been observed exactly $1+M_j^{t-1}$ times. Therefore

$$\frac{\sum_{s=1}^{J+t-1} X_{j_s}^s \mathbb{1}(j_s = j)}{N_j^{J+t-1}} = \overline{Y}_{j,1+M_j^{t-1}}.$$

On $\mathcal{E}$, we have

$$\left| \overline{Y}_{j,1+M_j^{t-1}} - r_j \right| \le \sqrt{\frac{2\log N}{1+M_j^{t-1}}}.$$

Adding the exploration term proves the stated bound. □

Lemma 14 (Cumulative-tracking discrepancy, Theorem 6 of Degenne et al. (2020))**.** *Let*

$$h_J = \sum_{i=2}^{J} \frac{1}{i}.$$

Under UCB-CT, it holds that for every $j \in [J]$ and every $t \in [T]$,

$$-h_J \le M_j^t - Q_j^t \le 1.$$

In particular,

$$Q_j^t \le M_j^t + h_J, \qquad Q_j^t - M_j^t \le h_J.$$

Proof of Lemma 3. The initialization phase is feasible because $c_j \geq 1$ for every $j \in [J]$.

Fix any round $n \in \{J+1, \ldots, N\}$. If server j is exhausted at the beginning of round n, then

$$N_j^{n-1} = c_j,$$

so its residual count equals

$$N_j^{n-1} - 1 = c_j - 1 = b_j.$$

On the other hand,

$$\sum_{s=J+1}^{n} q_j^s \leq (n-J)\bar{\mu}_j \leq T\bar{\mu}_j = b_j.$$

Therefore every exhausted server satisfies

$$\sum_{s=J+1}^{n} q_j^s - (N_j^{n-1} - 1) \leq 0.$$

Moreover,

$$\sum_{j=1}^{J} \left(\sum_{s=J+1}^{n} q_j^s - (N_j^{n-1} - 1) \right) = (n-J) - (n-1-J) = 1.$$

Hence at least one server has strictly positive discrepancy. Since every exhausted server has nonpositive discrepancy, any maximizer in the tracking step must be available. Thus $B_{j_n}^n > 0$ for all $n \in [N]$. □

Lemma 15 (Decomposition of regret)**.** *Under UCB-CT,*

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq J + Jh_J + \mathbb{E}[\Gamma_T].$$

Proof of Lemma 15. Let

$$\mathcal{R}_{\text{res}} = \sum_{t=1}^{T} X_{j_{J+t}}^{J+t}$$

be the reward collected in the residual phase. Since rewards are nonnegative,

$$R^* - \mathbb{E}\left[\sum_{n=1}^{N} X_{j_n}^n\right] \leq R^* - \mathbb{E}[\mathcal{R}_{\text{res}}].$$

By Lemma 12,

$$R^* - \mathbb{E}[\mathcal{R}_{\text{res}}] \leq J + R_{\text{res}}^* - \mathbb{E}[\mathcal{R}_{\text{res}}].$$

Also,

$$\mathbb{E}[\mathcal{R}_{\text{res}}] = \sum_{j=1}^{J} r_j \mathbb{E}[M_j^T],$$

so

$$R_{\text{res}}^* - \mathbb{E}[\mathcal{R}_{\text{res}}] = \mathbb{E}\left[R_{\text{res}}^* - \sum_{j=1}^{J} r_j M_j^T\right] = \mathbb{E}\left[\Gamma_T + \sum_{j=1}^{J} r_j (Q_j^T - M_j^T)\right].$$

By Lemma 14,

$$\sum_{j=1}^{J} r_j(Q_j^T - M_j^T) \le h_J \sum_{j=1}^{J} r_j \le J h_J.$$

Substituting this bound proves the lemma. □

C.2 Gap-dependent analysis

Throughout this subsection, assume $r_1 > r_2 > \cdots > r_J$. For the explicit logarithmic bound, let

$$\bar{S}_k = \sum_{i=1}^{k} \bar{\mu}_i, \qquad \widehat{j} = \max\{k \in \{0, \dots, J\} : \bar{S}_k \le 1\}, \qquad \bar{\delta} = 1 - \bar{S}_{\widehat{j}}.$$

If $\widehat{j} \ge 1$ and $\bar{\delta} > 0$, write

$$c = \widehat{j} + 1, \qquad Z = \{c+1, \dots, J\}.$$

If $\widehat{j} \ge 1$ and $\bar{\delta} = 0$, write

$$Z = \{\widehat{j} + 1, \dots, J\}.$$

Lemma 16 (Residual zero-mass target bound)**.** *Fix a residual zero-mass server j. Define*

$$u_j = \begin{cases} \left\lceil \dfrac{8 \log N}{(r_1 - r_j)^2} \right\rceil, & \widehat{j} = 0, \\ \left\lceil \dfrac{8 \log N}{(r_c - r_j)^2} \right\rceil, & \widehat{j} \ge 1 \text{ and } \bar{\delta} > 0, \\ \left\lceil \dfrac{8 \log N}{(r_{\widehat{j}} - r_j)^2} \right\rceil, & \widehat{j} \ge 1 \text{ and } \bar{\delta} = 0. \end{cases}$$

Then, on $\mathcal{E}$,

$$Q_j^T \le u_j + h_J.$$

Proof of Lemma 16. Let

$$\tau_j = \inf\{t \in [T] : M_j^t = u_j\},$$

with the convention $\tau_j = \infty$ if the set is empty. If $\tau_j = \infty$, then $M_j^T < u_j$, so Lemma 14 gives

$$Q_j^T \le M_j^T + h_J < u_j + h_J.$$

Assume now that $\tau_j < \infty$ and fix $t > \tau_j$. Then $M_j^{t-1} \ge u_j$, so on $\mathcal{E}$,

$$C_j^{J+t} \le r_j + 2\sqrt{\frac{2 \log N}{1 + M_j^{t-1}}} \le r_j + 2\sqrt{\frac{2 \log N}{1 + u_j}}.$$

By the definition of u_j, this implies

$$C_j^{J+t} < \begin{cases} r_1, & \widehat{j} = 0, \\ r_c, & \widehat{j} \ge 1 \text{ and } \bar{\delta} > 0, \\ r_{\widehat{j}}, & \widehat{j} \ge 1 \text{ and } \bar{\delta} = 0. \end{cases}$$

On the event $\mathcal{E}$, we also have $C_i^{J+t} \ge r_i$ for every i by Lemma 13. Hence:

- if $\widehat{j} = 0$, then $C_1^{J+t} > C_j^{J+t}$ and $\bar{\mu}_1 > 1$;
- if $\widehat{j} \geq 1$ and $\bar{\delta} > 0$, then $C_i^{J+t} > C_j^{J+t}$ for every $i = 1, \ldots, c$, and $\sum_{i=1}^{c} \bar{\mu}_i > 1$;
- if $\widehat{j} \geq 1$ and $\bar{\delta} = 0$, then $C_i^{J+t} > C_j^{J+t}$ for every $i = 1, \ldots, \widehat{j}$, and $\sum_{i=1}^{\widehat{j}} \bar{\mu}_i = 1$.

In each case, Lemma 10 applied to the LP implies that every optimal LP solution satisfies

$$q_j^{J+t} = 0.$$

Therefore $Q_j^T = Q_j^{\tau_j}$. Applying Lemma 14 at time τ_j yields

$$Q_j^T = Q_j^{\tau_j} \leq M_j^{\tau_j} + h_J = u_j + h_J.$$

This proves the claim. □

Lemma 17 (Residual critical target bound)**.** *Assume $\widehat{j} \geq 1$ and $\bar{\delta} > 0$. Let $c = \widehat{j} + 1$ and define*

$$u_c = \left\lceil \frac{8 \log N}{(r_{\widehat{j}} - r_c)^2} \right\rceil.$$

Then, on $\mathcal{E}$,

$$Q_c^T \leq \bar{\delta} T + u_c + h_J.$$

Proof of Lemma 17. Let

$$\tau_c = \inf\{t \in [T] : M_c^t = u_c\},$$

with the convention $\tau_c = \infty$ if the set is empty. If $\tau_c = \infty$, then $M_c^T < u_c$, so Lemma 14 gives

$$Q_c^T \leq M_c^T + h_J < u_c + h_J \leq \bar{\delta} T + u_c + h_J.$$

Assume now that $\tau_c < \infty$ and fix $t > \tau_c$. Then $M_c^{t-1} \geq u_c$, so on $\mathcal{E}$,

$$C_c^{J+t} \leq r_c + 2\sqrt{\frac{2 \log N}{1 + M_c^{t-1}}} \leq r_c + 2\sqrt{\frac{2 \log N}{1 + u_c}} < r_{\widehat{j}}.$$

For each $i = 1, \ldots, \widehat{j}$, Lemma 13 gives

$$C_i^{J+t} \geq r_i > r_{\widehat{j}} > C_c^{J+t}.$$

Therefore Lemma 11, applied to the LP with upper bounds $u_i = \bar{\mu}_i$, implies that every optimal LP solution satisfies

$$q_c^{J+t} \leq \bar{\delta}.$$

Consequently,

$$Q_c^T = Q_c^{\tau_c} + \sum_{t=\tau_c+1}^{T} q_c^{J+t} \leq (u_c + h_J) + \bar{\delta}(T - \tau_c) \leq \bar{\delta} T + u_c + h_J.$$

This proves the lemma. □

Lemma 18 (Target-selection loss on the good event)**.** *On the event $\mathcal{E}$, the following bounds hold.*
(i) If $\widehat{j} = 0$, then

$$\Gamma_T \le \sum_{j=2}^{J} (r_1 - r_j) A_j(N), \qquad A_j(N) = h_J + \left\lceil \frac{8 \log N}{(r_1 - r_j)^2} \right\rceil.$$

(ii) If $\widehat{j} \ge 1$ and $\bar{\delta} > 0$, then with $c = \widehat{j} + 1$ and $Z = \{c+1, \dots, J\}$,

$$\Gamma_T \le (r_1 - r_c) \left(A_c(N) + \sum_{j \in Z} A_j(N) \right) + \sum_{j \in Z} (r_c - r_j) A_j(N),$$

where

$$A_c(N) = h_J + \left\lceil \frac{8 \log N}{(r_{\widehat{j}} - r_c)^2} \right\rceil, \qquad A_j(N) = h_J + \left\lceil \frac{8 \log N}{(r_c - r_j)^2} \right\rceil, \quad j \in Z.$$

(iii) If $\widehat{j} \ge 1$ and $\bar{\delta} = 0$, then with $Z = \{\widehat{j} + 1, \dots, J\}$,

$$\Gamma_T \le (r_1 - r_{\widehat{j}}) \sum_{j \in Z} A_j(N) + \sum_{j \in Z} (r_{\widehat{j}} - r_j) A_j(N),$$

where

$$A_j(N) = h_J + \left\lceil \frac{8 \log N}{(r_{\widehat{j}} - r_j)^2} \right\rceil, \quad j \in Z.$$

Proof of Lemma 18. **Case 1: $\widehat{j} = 0$.** Then the optimal LP value is $R^*_{\text{res}} = T r_1$. Since $\sum_{j=1}^{J} Q_j^T = T$,

$$\Gamma_T = T r_1 - \sum_{j=1}^{J} r_j Q_j^T = \sum_{j=2}^{J} (r_1 - r_j) Q_j^T.$$

Applying Lemma 16 to each $j \ge 2$ yields the claim.

Case 2: $\widehat{j} \ge 1$ and $\bar{\delta} > 0$. Let $c = \widehat{j} + 1$ and $Z = \{c+1, \dots, J\}$. For $i \le \widehat{j}$, write

$$b_i = T \bar{\mu}_i = c_i - 1.$$

Then

$$R^*_{\text{res}} = \sum_{i=1}^{\widehat{j}} b_i r_i + \bar{\delta} T r_c.$$

A direct rearrangement gives

$$\Gamma_T = \sum_{i=1}^{\widehat{j}} (r_i - r_c)(b_i - Q_i^T) + \sum_{j \in Z} (r_c - r_j) Q_j^T.$$

Because $Q_i^T \le b_i$ for $i \le \widehat{j}$, the terms $b_i - Q_i^T$ are nonnegative. Moreover,

$$\sum_{i=1}^{\widehat{j}} (b_i - Q_i^T) = Q_c^T - \bar{\delta} T + \sum_{j \in Z} Q_j^T,$$

so

$$\Gamma_T \le (r_1 - r_c)\left(Q_c^T - \bar{\delta}T + \sum_{j\in Z} Q_j^T\right) + \sum_{j\in Z}(r_c - r_j)Q_j^T.$$

Applying Lemma 17 and Lemma 16 proves the stated bound.

Case 3: $\widehat{j} \ge 1$ and $\bar{\delta} = 0$. Let $Z = \{\widehat{j}+1, \dots, J\}$ and write $b_i = T\bar{\mu}_i$ for $i \le \widehat{j}$. Then

$$R^*_{\mathrm{res}} = \sum_{i=1}^{\widehat{j}} b_i r_i.$$

Again by direct rearrangement,

$$\Gamma_T = \sum_{i=1}^{\widehat{j}} (r_i - r_{\widehat{j}})(b_i - Q_i^T) + \sum_{j\in Z}(r_{\widehat{j}} - r_j)Q_j^T.$$

Since

$$\sum_{i=1}^{\widehat{j}}(b_i - Q_i^T) = \sum_{j\in Z} Q_j^T,$$

we obtain

$$\Gamma_T \le (r_1 - r_{\widehat{j}}) \sum_{j\in Z} Q_j^T + \sum_{j\in Z}(r_{\widehat{j}} - r_j)Q_j^T.$$

Applying Lemma 16 to each $j \in Z$ yields the claim. □

Proof of Theorem 1. By Lemma 15, it suffices to bound $\mathbb{E}[\Gamma_T]$. Since $0 \le \Gamma_T \le T$, Lemma 13 gives

$$\mathbb{E}[\Gamma_T] \le \mathbb{E}[\Gamma_T \mathbb{1}(\mathcal{E})] + T\mathbb{P}(\mathcal{E}^c) \le \mathbb{E}[\Gamma_T \mathbb{1}(\mathcal{E})] + \frac{2J}{N^2}.$$

On the event $\mathcal{E}$, Lemma 18 gives exactly the corresponding case-wise bound for Γ_T. Substituting that bound into Lemma 15 proves all three cases. □

C.3 Worst-case analysis

Lemma 19. *For every $j \in [J]$,*

$$\sum_{t=1}^{T} \frac{q_j^{J+t}}{\sqrt{1 + M_j^{t-1}}} \le 2(h_J + 2)\sqrt{M_j^T + 1}.$$

Consequently,

$$\sum_{t=1}^{T}\sum_{j=1}^{J} \frac{q_j^{J+t}}{\sqrt{1 + M_j^{t-1}}} \le 2(h_J + 2)\sqrt{JN}.$$

Proof of Lemma 19. Fix $j \in [J]$ and let $m_j = M_j^T$. Let

$$0 = \tau_{j,0} < \tau_{j,1} < \cdots < \tau_{j,m_j} \le T$$

be the residual rounds at which server j is selected, and set $\tau_{j,m_j+1} = T$. Then $M_j^{t-1} = m$ for all

$$t = \tau_{j,m} + 1, \dots, \tau_{j,m+1}, \qquad m = 0, \dots, m_j.$$

Hence

$$\sum_{t=1}^{T} \frac{q_j^{J+t}}{\sqrt{1 + M_j^{t-1}}} = \sum_{m=0}^{m_j} \frac{Q_j^{\tau_{j,m+1}} - Q_j^{\tau_{j,m}}}{\sqrt{m+1}}.$$

By Lemma 14,

$$M_j^{\tau_{j,m}} - Q_j^{\tau_{j,m}} \le 1, \qquad M_j^{\tau_{j,m+1}} - Q_j^{\tau_{j,m+1}} \ge -h_J,$$

so

$$Q_j^{\tau_{j,m+1}} - Q_j^{\tau_{j,m}} \le h_J + 2.$$

Therefore,

$$\sum_{t=1}^{T} \frac{q_j^{J+t}}{\sqrt{1 + M_j^{t-1}}} \le (h_J + 2) \sum_{m=0}^{m_j} \frac{1}{\sqrt{m+1}} \le 2(h_J + 2)\sqrt{m_j + 1}.$$

Summing over j and applying Cauchy–Schwarz yields

$$\sum_{j=1}^{J} \sqrt{M_j^T + 1} \le \sqrt{J \sum_{j=1}^{J} (M_j^T + 1)} = \sqrt{J(T+J)} = \sqrt{JN},$$

because $\sum_{j=1}^{J} M_j^T = T$. This proves the lemma. □

Proof of Theorem 2. By Lemma 15, it suffices to bound $\mathbb{E}[\Gamma_T]$. We first work on the event $\mathcal{E}$ from Lemma 13. Let $\boldsymbol{q}^\star$ be an optimal solution to the LP, so that

$$R_{\text{res}}^* = T \sum_{j=1}^{J} q_j^\star r_j.$$

For each residual round $t \in [T]$, the vector $\boldsymbol{q}^{J+t}$ maximizes $\sum_{j=1}^{J} q_j C_j^{J+t}$ over the residual feasible set, hence

$$\sum_{j=1}^{J} q_j^\star C_j^{J+t} \le \sum_{j=1}^{J} q_j^{J+t} C_j^{J+t}.$$

Therefore

$$\sum_{j=1}^{J} q_j^\star r_j - \sum_{j=1}^{J} q_j^{J+t} r_j \le \sum_{j=1}^{J} q_j^{J+t} (C_j^{J+t} - r_j) + \sum_{j=1}^{J} q_j^\star (r_j - C_j^{J+t}).$$

On $\mathcal{E}$, Lemma 13 gives $r_j \le C_j^{J+t}$ for every j, so the second term is nonpositive. The same lemma also yields

$$C_j^{J+t} - r_j \le 2\sqrt{\frac{2 \log N}{1 + M_j^{t-1}}}.$$

Hence, on $\mathcal{E}$,

$$\sum_{j=1}^{J} q_j^\star r_j - \sum_{j=1}^{J} q_j^{J+t} r_j \le 2\sqrt{2 \log N} \sum_{j=1}^{J} \frac{q_j^{J+t}}{\sqrt{1 + M_j^{t-1}}}.$$

Summing over $t = 1, \ldots, T$ and applying Lemma 19 yields

$$\Gamma_T = R^*_{\text{res}} - \sum_{j=1}^{J} r_j Q_j^T \leq 4(h_J + 2)\sqrt{2JN \log N} \qquad \text{on } \mathcal{E}.$$

Since also $0 \leq \Gamma_T \leq T$, we obtain

$$\mathbb{E}[\Gamma_T] \leq 4(h_J + 2)\sqrt{2JN \log N} + T\mathbb{P}(\mathcal{E}^c) \leq 4(h_J + 2)\sqrt{2JN \log N} + \frac{2J}{N^2},$$

where the last step uses Lemma 13. Substituting this bound into Lemma 15 proves the theorem. □

D Proofs for TS-PS

D.1 Proof of admissibility

Proof of Lemma 4. If $B^n_{\widehat{j}_n} > 0$, then $j_n = \widehat{j}_n$, and the claim is immediate. If $B^n_{\widehat{j}_n} = 0$, then the algorithm selects

$$j_n \in \operatorname*{argmax}_{j \in [J]} \left\{ np_j^n - N_j^{n-1} \right\}.$$

Since $\boldsymbol{p}^n$ is feasible for the LP, the same argument as in the proof of Lemma 7 shows that this tracking step always selects an available server. Hence $B^n_{j_n} > 0$ in both cases. □

D.2 Proof of Bayes Regret

Proof of Theorem 3. This is the special case $M = 1$ of Theorem 4. Indeed, when $M = 1$, the customer-level LP (4) reduces to the homogeneous LP (1), so $\psi(1) = R^*$. Moreover, Algorithm 4 coincides with Algorithm 3. Therefore Theorem 4 yields

$$\mathbb{E}_{\boldsymbol{r} \sim \mathbb{Q}^1} \left[R^* - \mathbb{E}\left[\sum_{n=1}^{N} X^n_{j_n} \right] \right] \leq \left(6\sqrt{J} + 12\sqrt{J \log J} + 3(J-1)\sqrt{J} \right) \sqrt{N}.$$ □

E Proofs for the Heterogeneous Case

Proof of Proposition 3. For each type $i \in [I]$, let

$$M_i = \sum_{m=1}^{M} \mathbb{1}(z(m) = i), \qquad \widehat{\rho}_i = \frac{M_i}{M}.$$

Because customers of the same type have identical reward vectors, there exists an optimal solution of (4) that assigns the same allocation vector to all customers of the same type. Therefore, (4) has

the same optimal value as

$$\begin{aligned} \max_{\{p_{i,j}\}} \quad & N \sum_{i=1}^{I} \sum_{j=1}^{J} \widehat{\rho}_i p_{i,j} r_{i,j} \\ \text{s.t.} \quad & \sum_{i=1}^{I} \widehat{\rho}_i p_{i,j} \le \mu_j, \qquad j \in [J], \\ & \sum_{j=1}^{J} p_{i,j} = 1, \quad i \in [I], \qquad p_{i,j} \ge 0, \quad i \in [I],\ j \in [J]. \end{aligned} \tag{10}$$

This is exactly the LP (3) with ρ_i replaced by $\widehat{\rho}_i$.

By the strong law of large numbers, $\widehat{\rho}_i \to \rho_i$ almost surely for every $i \in [I]$. The feasible set of (10) is compact, and both the objective and the constraints depend continuously on $\widehat{\rho}$. Hence the optimal value is a continuous function of $\widehat{\rho}$, and therefore

$$\psi(M) \to \psi^* \qquad \text{a.s.}$$

Moreover, for every feasible solution of (4),

$$0 \le \frac{N}{M} \sum_{m=1}^{M} \sum_{j=1}^{J} p_j(m) r_j(m) \le \frac{N}{M} \sum_{m=1}^{M} \sum_{j=1}^{J} p_j(m) = N.$$

Thus $0 \le \psi(M) \le N$ almost surely for every M. The convergence of expectations then follows from dominated convergence. □

Proof of Lemma 5. At every decision epoch, the algorithm either assigns the queried server $\widehat{j}_n(m)$, which is feasible by construction when $B_{\widehat{j}_n(m)} > 0$, or it selects a server from the set $\{j : B_j > 0\}$. Hence it never assigns a task to an exhausted server. □

For the analysis in this subsection, let

$$\mathcal{H}_{n-1} = \sigma\big(j_s(m), X^s_{j_s(m)}(m) : m \in [M],\ s = 1, \dots, n-1\big)$$

denote the realized-assignment history through stage $n-1$. The posterior samples $\{\boldsymbol{S}^n(m)\}_{m=1}^M$ are drawn from the joint posterior of the true reward vectors given $\mathcal{H}_{n-1}$. The sampled queries $\widehat{j}_n(m)$ are used only to randomize the intended assignment. If a query is repaired, the unavailable queried reward is not observed; the posterior is updated from $(j_n(m), X^n_{j_n(m)}(m))$ only.

Lemma 20 (Auxiliary bounds under realized feedback)**.** *Assume $J \ge 2$ and fix a customer $m \in [M]$. Let*

$$N_j^{n-1}(m) = \sum_{s=1}^{n-1} \mathbb{1}(j_s(m) = j)$$

be the number of realized assignments of customer m to server j up to stage $n-1$, and define

$$\widehat{r}_j^n(m) = \begin{cases} \dfrac{\sum_{s=1}^{n-1} X^s_{j_s(m)}(m) \mathbb{1}(j_s(m) = j)}{N_j^{n-1}(m)}, & N_j^{n-1}(m) \ge 1, \\ 0, & N_j^{n-1}(m) = 0. \end{cases}$$

Let

$$U_j^n(m) = \begin{cases} 1, & N_j^{n-1}(m) = 0, \\ \min\left\{1, \widehat{r}_j^n(m) + \sqrt{\dfrac{\log_+\left(\dfrac{JN}{N_j^{n-1}(m)}\right)}{N_j^{n-1}(m)}}\right\}, & N_j^{n-1}(m) \geq 1, \end{cases}$$

where $\log_+(x) = \max\{\log x, 0\}$. *Then* $U_j^n(m)$ *is* $\mathcal{H}_{n-1}$*-measurable and*

$$\mathbb{E}[(r_j(m) - U_j^n(m))^+] \leq \frac{6}{\sqrt{JN}}, \qquad j \in [J],\ n \in [N],$$

and

$$\sum_{n=1}^{N}\sum_{j=1}^{J} \mathbb{E}\left[\mathbb{1}(j_n(m) = j)(U_j^n(m) - r_j(m))\right] \leq 12\sqrt{JN\log J}.$$

Proof of Lemma 20. Fix customer m and consider the pre-decision filtration that contains $\mathcal{H}_{n-1}$, the stage-n posterior samples and queries, and the assignments and observations of customers processed before m in stage n. The realized action $j_n(m)$ is measurable with respect to this filtration, whereas, conditional on $\boldsymbol{r}(m)$, the current potential rewards $\{X_j^n(m) : j \in [J]\}$ are independent of it. Consequently, the realized assignments and observed rewards of customer m form an adaptive J-armed bandit process with horizon N in this enlarged filtration. The action may depend on the global capacity state and on the posterior samples of all customers, but it is selected before $X_{j_n(m)}^n(m)$ is observed.

Specialize Lemmas EC.1 and EC.2 of Ferreira et al. (2018) to $T = N$, $K = J$, and the realized action counts $N_j^{n-1}(m)$. Their UCB is exactly $U_j^n(m)$ above. Lemma EC.1 gives the pointwise underestimation bound. Lemma EC.2 applies to any predictable action rule and gives the cumulative optimism bound weighted by the pre-decision action probabilities. By iterated expectation, that weighted expression is equal to the indicator form displayed above. This proves both claims. □

Lemma 21 (Number of feasibility repairs)**.** *Let*

$$D = \sum_{m=1}^{M}\sum_{n=1}^{N} \mathbb{1}(j_n(m) \neq \widehat{j}_n(m))$$

be the number of tasks whose sampled server is unavailable and hence repaired. Under Algorithm 4,

$$\mathbb{E}[D] \leq (J-1)\sqrt{JNM}.$$

Proof of Lemma 21. Flatten the customer–stage pairs (n, m) in lexicographic order and recall that $c_j = \mu_j NM$ is server j's capacity. Let

$$K_j = \sum_{m=1}^{M}\sum_{n=1}^{N} \mathbb{1}(\widehat{j}_n(m) = j)$$

be the total number of queries to server j. We first relate D to the query overflows $(K_j - c_j)^+$.

For every repaired task, draw a directed edge from the unavailable queried server to the available server that actually receives the task. Let τ_j be the lexicographic epoch at which server j receives its c_jth realized assignment, with $\tau_j = \infty$ if it never becomes exhausted. If a repair creates an edge $j \to k$ at epoch t, then j was exhausted before t, whereas k was available immediately before the assignment at t. Hence $\tau_j < t \le \tau_k$, so every repair edge points from an earlier depletion epoch to a later one. The directed multigraph of repairs is therefore acyclic.

Let O_j and I_j be the outgoing and incoming repair counts at node j, let H_j be the number of queries to j that are honored, and let A_j be the total realized assignment count at server j. Then

$$K_j = H_j + O_j, \qquad A_j = H_j + I_j, \qquad O_j - I_j = K_j - A_j.$$

If $O_j = 0$, then $O_j - I_j = -I_j \le 0$ and $K_j \le A_j \le c_j$. If $O_j > 0$, server j is exhausted by the end of the batch, so $A_j = c_j$. It follows in both cases that

$$(O_j - I_j)^+ = (K_j - c_j)^+.$$

Thus the total positive divergence of the repair flow is exactly $\sum_j (K_j - c_j)^+$. An acyclic integral flow decomposes into directed paths from positive-divergence nodes to negative-divergence nodes. Every such path visits each of the J servers at most once and therefore has at most $J - 1$ edges. Consequently,

$$D = \sum_{j=1}^{J} O_j \le (J-1) \sum_{j=1}^{J} (K_j - c_j)^+.$$

It remains to control the total query overflow. Let $\boldsymbol{e}_j$ denote the jth coordinate vector and define the vector martingale increment

$$\boldsymbol{Z}_{n,m} = \boldsymbol{e}_{\widehat{j}_n(m)} - \boldsymbol{p}^n(m).$$

Reveal the queries in lexicographic order, conditional on the stage history and the posterior samples. Then $\mathbb{E}[\boldsymbol{Z}_{n,m} \mid \cdot] = \boldsymbol{0}$. Since the LP solution is feasible at every stage,

$$P_j := \sum_{n=1}^{N} \sum_{m=1}^{M} p_j^n(m) \le \mu_j NM = c_j.$$

Writing $\boldsymbol{Z} = \sum_{n,m} \boldsymbol{Z}_{n,m}$, we have $K_j - c_j \le K_j - P_j = Z_j$, and therefore

$$\sum_{j=1}^{J} (K_j - c_j)^+ \le \sum_{j=1}^{J} (Z_j)^+ \le \sqrt{J}\, \|\boldsymbol{Z}\|_2.$$

Martingale orthogonality and the fact that a categorical draw has conditional second moment

$$\mathbb{E}[\|\boldsymbol{Z}_{n,m}\|_2^2 \mid \cdot] = 1 - \|\boldsymbol{p}^n(m)\|_2^2 \le 1$$

give

$$\mathbb{E}[\|\boldsymbol{Z}\|_2^2] \le NM.$$

Applying Cauchy–Schwarz and then the pathwise flow bound yields

$$\mathbb{E}[D] \le (J-1)\sqrt{J}\,\mathbb{E}[\|\boldsymbol{Z}\|_2] \le (J-1)\sqrt{JNM}.$$

□

Proof of Theorem 4. Let

$$\mathbf{r} = (\boldsymbol{r}(1), \dots, \boldsymbol{r}(M)) \qquad \text{and} \qquad \mathbf{S}^n = (\boldsymbol{S}^n(1), \dots, \boldsymbol{S}^n(M))$$

denote the true customer-reward vectors and the joint posterior sample at stage n. Let $\widehat{\jmath}_n(m)$ be the sampled server and $j_n(m)$ the server that actually processes the task. Although the reward $X^n_{\widehat{\jmath}_n(m)}(m)$ is unobserved when a repair occurs, it is a well-defined potential outcome and may be used as an analytical coupling variable. Since rewards lie in $[0,1]$, pathwise

$$\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{j_n(m)}(m) \ge \sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{\widehat{\jmath}_n(m)}(m) - D.$$

Therefore

$$\mathrm{BayesRegret}_M(N) \le T_1 + \frac{\mathbb{E}[D]}{M},$$

where

$$T_1 = \mathbb{E}\left[\psi(M) - \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{\widehat{\jmath}_n(m)}(m)\right].$$

Let $\{\boldsymbol{p}^\star(m)\}_{m=1}^M$ be the optimizer of (4) under $\mathbf{r}$, selected by the fixed tie-breaking rule. Then

$$\psi(M) = \frac{1}{M}\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{j=1}^{J} p_j^\star(m) r_j(m).$$

Conditional on $\mathcal{H}_{n-1}$, $\mathbf{S}^n$, and $\mathbf{r}$, the query $\widehat{\jmath}_n(m)$ is sampled from $\boldsymbol{p}^n(m)$, so

$$\mathbb{E}\big[X^n_{\widehat{\jmath}_n(m)}(m) \mid \mathcal{H}_{n-1}, \mathbf{S}^n, \mathbf{r}\big] = \sum_{j=1}^{J} p_j^n(m) r_j(m).$$

Thus

$$T_1 = \frac{1}{M}\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{j=1}^{J} \mathbb{E}\big[(p_j^\star(m) - p_j^n(m)) r_j(m)\big].$$

Let Π denote the deterministic LP solution map. By posterior sampling,

$$\mathbf{S}^n \mid \mathcal{H}_{n-1} \overset{d}{=} \mathbf{r} \mid \mathcal{H}_{n-1},$$

so $\Pi(\mathbf{S}^n)$ and $\Pi(\mathbf{r})$ have the same conditional distribution. Because $U_j^n(m)$ is $\mathcal{H}_{n-1}$-measurable,

$$\mathbb{E}\left[\sum_{m,j} p_j^\star(m) U_j^n(m) \middle| \mathcal{H}_{n-1}\right] = \mathbb{E}\left[\sum_{m,j} p_j^n(m) U_j^n(m) \middle| \mathcal{H}_{n-1}\right].$$

Adding and subtracting $U_j^n(m)$ yields

$$T_1 = \frac{1}{M}\sum_{n,m,j}\mathbb{E}[p_j^\star(m)(r_j(m) - U_j^n(m))] + \frac{1}{M}\sum_{n,m,j}\mathbb{E}[p_j^n(m)(U_j^n(m) - r_j(m))].$$

The first term is bounded by $6\sqrt{JN}$ using Lemma 20, because $0 \le p_j^\star(m) \le 1$.

For the second term, conditional on $\mathcal{H}_{n-1}$, $\mathbf{S}^n$, and $\mathbf{r}$,

$$\sum_{j=1}^{J} p_j^n(m)(U_j^n(m) - r_j(m)) = \mathbb{E}[U^n_{\widehat{j}_n(m)}(m) - r_{\widehat{j}_n(m)}(m) \mid \cdot].$$

If no repair occurs, $\widehat{j}_n(m) = j_n(m)$. If a repair occurs, both $U_j^n(m)$ and $r_j(m)$ lie in $[0,1]$, so

$$U^n_{\widehat{j}_n(m)}(m) - r_{\widehat{j}_n(m)}(m) \le U^n_{j_n(m)}(m) - r_{j_n(m)}(m) + 2\mathbb{1}(j_n(m) \ne \widehat{j}_n(m)).$$

Summing, applying iterated expectation, and then using Lemma 20 customer by customer gives

$$\frac{1}{M}\sum_{n,m,j}\mathbb{E}[p_j^n(m)(U_j^n(m) - r_j(m))] \le 12\sqrt{JN\log J} + \frac{2\mathbb{E}[D]}{M}.$$

Consequently,

$$T_1 \le 6\sqrt{JN} + 12\sqrt{JN\log J} + \frac{2\mathbb{E}[D]}{M}.$$

Combining this bound with the coupling loss and Lemma 21, we obtain

$$\begin{aligned}\mathrm{BayesRegret}_M(N) &\le 6\sqrt{JN} + 12\sqrt{JN\log J} + \frac{3\mathbb{E}[D]}{M} \\ &\le \left(6\sqrt{J} + 12\sqrt{J\log J}\right)\sqrt{N} + 3(J-1)\sqrt{\frac{JN}{M}},\end{aligned}$$

which proves the theorem using only realized-assignment feedback. □

Proof of Corollary 1. By adding and subtracting $\psi(M)$, we obtain

$$\mathbb{E}\left[\psi^* - \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{j_n(m)}(m)\right] = \psi^* - \mathbb{E}[\psi(M)] + \mathbb{E}\left[\psi(M) - \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{j_n(m)}(m)\right].$$

By Proposition 3,

$$\psi^* - \mathbb{E}[\psi(M)] \to 0 \qquad \text{as } M \to \infty.$$

By Theorem 4,

$$\mathbb{E}\left[\psi(M) - \frac{1}{M}\sum_{m=1}^{M}\sum_{n=1}^{N} X^n_{j_n(m)}(m)\right] \le \left(6\sqrt{J} + 12\sqrt{J\log J}\right)\sqrt{N} + 3(J-1)\sqrt{\frac{JN}{M}}.$$

Taking the upper limit as $M \to \infty$ proves the result. □

F Analysis of Accelerated EM

The probability is taken over the latent customer types, the Bernoulli rewards, and the random initialization in Algorithm 5. Recall that $z(1), \ldots, z(H)$ are i.i.d. and $\mathbb{P}(z(h) = i) = \rho_i$ for every $h \in [H]$ and every $i \in [I]$. For each $i \in [I]$, let $\mathcal{H}_i = \{h \in [H] : z(h) = i\}$ and let $H_i = |\mathcal{H}_i|$. We refer to $\mathcal{H}_i$ as cluster i in the clustering analysis. Let $\rho_{\min} = \min_{1 \le i \le I} \rho_i$.

We first define

$$\phi \triangleq \min_{1 \le i \le I} \min_{\substack{1 \le k \le I \\ k \ne i}} \frac{1}{2} \sum_{j=1}^{J} \mathrm{KL}(r_{i,j}, r_{k,j}).$$

Under Assumption 1, we have $\phi > 0$. For all $1 \le i \le I$ and $1 \le k \le I$ with $k \ne i$, define

$$\begin{aligned}
A_{i,k} &\triangleq \frac{1}{2} \sum_{j=1}^{J} \mathrm{KL}(r_{i,j}, r_{k,j}), \\
B_{i,k} &\triangleq \sum_{j=1}^{J} \left(\frac{1}{\sqrt{2}} \left| \log \frac{r_{i,j}(1 - r_{k,j})}{r_{k,j}(1 - r_{i,j})} \right| + 2\sqrt{2} \left(\frac{1 + r_{i,j}}{1 - r_{i,j}} + \frac{1 + r_{i,j}}{1 - r_{k,j}} + 1 + \frac{r_{i,j}}{r_{k,j}} \right) \right), \\
C_{i,k} &\triangleq 2 \sum_{j=1}^{J} \left(\frac{1}{r_{i,j}} + \frac{1}{r_{k,j}} + \frac{1}{1 - r_{k,j}} + \frac{1}{1 - r_{i,j}} \right).
\end{aligned}$$

Define

$$\begin{aligned}
\delta_1 &\triangleq \max_{1 \le i \le I, 1 \le j \le J} \frac{25}{8 r_{i,j}^2}, \\
\delta_2 &\triangleq \max_{1 \le i \le I, 1 \le j \le J} \frac{25}{8 (1 - r_{i,j})^2}, \\
\delta_3 &\triangleq \max_{\substack{1 \le i \le I, 1 \le k \le I \\ k \ne i}} \left(\frac{B_{i,k} + \sqrt{B_{i,k}^2 + 4 A_{i,k} C_{i,k}}}{2 A_{i,k}} \right)^2, \\
\delta_4 &\triangleq \max_{\substack{1 \le i \le I, 1 \le k \le I \\ k \ne i}} \frac{32 \left(\sum_{j=1}^{J} |r_{i,j} - r_{k,j}| \right)^2}{\left(\sum_{j=1}^{J} |r_{i,j} - r_{k,j}|^2 \right)^2}.
\end{aligned}$$

We eventually define

$$a \triangleq \max\{\delta_1, \delta_2, \delta_3, \delta_4\}.$$

Lemma 22 (Empirical type frequencies)**.** *With probability at least* $1 - 2Ie^{-H\rho_{\min}/48}$*, we have*

$$\frac{3}{4}\rho_i \le \frac{H_i}{H} \le \frac{5}{4}\rho_i$$

for all $i \in [I]$*.*

Proof. For each $i \in [I]$, we have $H_i \sim \mathrm{Binomial}(H, \rho_i)$. By the multiplicative Chernoff bound,

$$\mathbb{P}\left(H_i \le \frac{3}{4} H \rho_i \right) \le e^{-H\rho_i/32},$$

and

$$\mathbb{P}\left(H_i \geq \frac{5}{4}H\rho_i\right) \leq e^{-H\rho_i/48}.$$

A union bound over $i \in [I]$ completes the proof. □

For each type $i \in [I]$, define

$$\mathcal{C}_i \triangleq \{X(\ell) : 1 \leq \ell \leq L, X(\ell) \in \mathcal{H}_i\}.$$

Lemma 23 (Initialization)**.** *With probability at least* $1 - 2Ie^{-L\rho_{\min}/8}$*, we have* $1 \leq |\mathcal{C}_i| \leq \frac{3}{2}L\rho_i$ *for all* $i \in [I]$*.*

Proof. By exchangeability, the vector $(z(X(1)), \ldots, z(X(L)))$ has the same distribution as $(z(1), \ldots, z(L))$. Hence, for each $i \in [I]$, the count $|\mathcal{C}_i|$ is distributed as Binomial(L, ρ_i). Therefore,

$$\mathbb{P}(|\mathcal{C}_i| = 0) = (1-\rho_i)^L \leq e^{-L\rho_i} \leq e^{-L\rho_{\min}/8}.$$

Also,

$$\mathbb{P}\left(|\mathcal{C}_i| \geq \frac{3}{2}L\rho_i\right) \leq \exp\left(-\frac{(L\rho_i/2)^2}{2L\rho_i(1-\rho_i)}\right) \leq e^{-L\rho_i/8} \leq e^{-L\rho_{\min}/8}.$$

A union bound over $i \in [I]$ completes the proof. □

Define the event

$$\mathcal{E}_0 \triangleq \left\{\left|\frac{\alpha_j(h)}{n_j(h)} - r_{z(h),j}\right| \leq \frac{1}{\sqrt{2a}} \text{ for all } h \in [H], j \in [J]\right\}.$$

For every $h \in \mathcal{H}_i$ and every $j \in [J]$, Hoeffding's inequality and Assumption 2 imply

$$\mathbb{P}\left(\left|\frac{\alpha_j(h)}{n_j(h)} - r_{i,j}\right| > \frac{1}{\sqrt{2a}}\right) \leq \mathbb{P}\left(\left|\frac{\alpha_j(h)}{n_j(h)} - r_{i,j}\right| > \sqrt{\frac{\log N}{2n_j(h)}}\right) \leq \frac{2}{N}.$$

Therefore,

$$\mathbb{P}(\mathcal{E}_0) \geq 1 - \frac{2HJ}{N}.$$

Lemma 24 (First round soft assignments)**.** *On the event* $\mathcal{E}_0$*, we have* $p_\ell^{(1)}(\boldsymbol{\alpha}(h)) \geq N^{a\phi} p_m^{(1)}(\boldsymbol{\alpha}(h))$ *for all* $h \in \mathcal{H}_i$*, all* $X(\ell) \in \mathcal{C}_i$*, all* $X(m) \in \mathcal{C}_k$*, and all* $i \neq k$*.*

Proof. Fix $h \in \mathcal{H}_i$, $X(\ell) \in \mathcal{C}_i$, and $X(m) \in \mathcal{C}_k$ with $k \neq i$. Recall that $r_{\ell,j}^{(0)} = \frac{\alpha_j(X(\ell))}{n_j(X(\ell))}$. On the event $\mathcal{E}_0$, we have

$$\left|r_{\ell,j}^{(0)} - r_{i,j}\right| \leq \frac{1}{\sqrt{2a}} \quad \text{and} \quad \left|r_{m,j}^{(0)} - r_{k,j}\right| \leq \frac{1}{\sqrt{2a}}$$

for all $j \in [J]$. Since $a \geq \max\{\delta_1, \delta_2\}$, these bounds imply

$$\left|\log \frac{r_{\ell,j}^{(0)}}{r_{i,j}}\right| \leq \sqrt{\frac{8}{ar_{i,j}^2}}, \qquad \left|\log \frac{1-r_{\ell,j}^{(0)}}{1-r_{i,j}}\right| \leq \sqrt{\frac{8}{a(1-r_{i,j})^2}},$$

and

$$\left|\log \frac{r_{m,j}^{(0)}}{r_{k,j}}\right| \le \sqrt{\frac{8}{a r_{k,j}^2}}, \qquad \left|\log \frac{1-r_{m,j}^{(0)}}{1-r_{k,j}}\right| \le \sqrt{\frac{8}{a(1-r_{k,j})^2}}.$$

Because the initialization weights are equal, we have

$$\log \frac{p_\ell^{(1)}(\boldsymbol{\alpha}(h))}{p_m^{(1)}(\boldsymbol{\alpha}(h))} = \sum_{j=1}^{J} \alpha_j(h) \log \frac{r_{\ell,j}^{(0)}}{r_{m,j}^{(0)}} + \sum_{j=1}^{J} (n_j(h) - \alpha_j(h)) \log \frac{1-r_{\ell,j}^{(0)}}{1-r_{m,j}^{(0)}}.$$

Rearranging around the true parameters gives

$$\log \frac{p_\ell^{(1)}(\boldsymbol{\alpha}(h))}{p_m^{(1)}(\boldsymbol{\alpha}(h))} = \sum_{j=1}^{J} n_j(h) \Bigg[\mathrm{KL}(r_{i,j}, r_{k,j}) + \left(\frac{\alpha_j(h)}{n_j(h)} - r_{i,j}\right) \log \frac{r_{i,j}(1-r_{k,j})}{r_{k,j}(1-r_{i,j})} \quad + \log \frac{1-r_{\ell,j}^{(0)}}{1-r_{i,j}} \frac{1-r_{k,j}}{1-r_{m,j}^{(0)}}$$
$$+ \frac{\alpha_j(h)}{n_j(h)} \left(\log \frac{r_{\ell,j}^{(0)}}{r_{i,j}} - \log \frac{r_{m,j}^{(0)}}{r_{k,j}} + \log \frac{1-r_{m,j}^{(0)}}{1-r_{k,j}} - \log \frac{1-r_{\ell,j}^{(0)}}{1-r_{i,j}} \right) \Bigg].$$

On $\mathcal{E}_0$, we also have

$$\frac{\alpha_j(h)}{n_j(h)} \le r_{i,j} + \frac{1}{\sqrt{2a}}.$$

Using $n_j(h) \ge a \log N$, the bounds above, and the definitions of $A_{i,k}$, $B_{i,k}$, and $C_{i,k}$, we obtain

$$\log \frac{p_\ell^{(1)}(\boldsymbol{\alpha}(h))}{p_m^{(1)}(\boldsymbol{\alpha}(h))} \ge a \log N \left(2A_{i,k} - \frac{B_{i,k}}{\sqrt{a}} - \frac{C_{i,k}}{a} \right).$$

Since $a \ge \delta_3$, we have

$$2A_{i,k} - \frac{B_{i,k}}{\sqrt{a}} - \frac{C_{i,k}}{a} \ge A_{i,k} \ge \phi.$$

Therefore,

$$\log \frac{p_\ell^{(1)}(\boldsymbol{\alpha}(h))}{p_m^{(1)}(\boldsymbol{\alpha}(h))} \ge a\phi \log N.$$

Exponentiating both sides completes the proof. □

Lemma 25. *Choose ρ_T satisfying*

$$\rho_T \le \overline{\rho} \triangleq \frac{1 - LN^{-a\phi}}{2L}.$$

Let

$$\mathcal{C}_i^P \triangleq \{X(\ell) \in \mathcal{C}_i : \rho_\ell^{(1)} \ge \rho_T\}.$$

On the event in Lemma 22, the event in Lemma 23, and the event $\mathcal{E}_0$, we have that $\mathcal{C}_i^P$ is nonempty for all $i \in [I]$.

Proof. Fix $i \in [I]$ and fix $h \in \mathcal{H}_i$. By Lemma 23, the set $\mathcal{C}_i$ is nonempty. Lemma 24 then implies

$$p_m^{(1)}(\boldsymbol{\alpha}(h)) \le N^{-a\phi}$$

for every $X(m) \in \mathcal{C}_k$ with $k \neq i$. Therefore,

$$\sum_{k \neq i} \sum_{X(m) \in \mathcal{C}_k} p_m^{(1)}(\boldsymbol{\alpha}(h)) \leq LN^{-a\phi},$$

and hence

$$\sum_{X(\ell) \in \mathcal{C}_i} p_\ell^{(1)}(\boldsymbol{\alpha}(h)) \geq 1 - LN^{-a\phi}.$$

Now,

$$\sum_{X(\ell) \in \mathcal{C}_i} \rho_\ell^{(1)} = \frac{1}{H} \sum_{h'=1}^{H} \sum_{X(\ell) \in \mathcal{C}_i} p_\ell^{(1)}(\boldsymbol{\alpha}(h')) \geq \frac{1}{H} \sum_{h' \in \mathcal{H}_i} \sum_{X(\ell) \in \mathcal{C}_i} p_\ell^{(1)}(\boldsymbol{\alpha}(h')) \geq \frac{H_i}{H}(1 - LN^{-a\phi}).$$

By Lemma 22, we have $\frac{H_i}{H} \geq \frac{3}{4}\rho_i$. By Lemma 23, we also have $|\mathcal{C}_i| \leq \frac{3}{2}L\rho_i$. Therefore,

$$\frac{1}{|\mathcal{C}_i|} \sum_{X(\ell) \in \mathcal{C}_i} \rho_\ell^{(1)} \geq \frac{(3/4)\rho_i(1 - LN^{-a\phi})}{(3/2)L\rho_i} = \frac{1 - LN^{-a\phi}}{2L} = \overline{\rho} \geq \rho_T.$$

By the pigeonhole principle, at least one point in $\mathcal{C}_i$ has first-round mixing weight at least ρ_T. This proves that $\mathcal{C}_i^P$ is nonempty. □

Lemma 26 (First round center estimates)**.** *Choose ρ_T satisfying $\rho_T \leq \overline{\rho}$ and*

$$\rho_T \geq \underline{\rho} \triangleq \max_{1 \leq i \leq I, 1 \leq j \leq J} \frac{1 - \frac{3}{4}\rho_i}{N^{a\phi}} \left(\sqrt{2a} \max_{1 \leq k \leq I} |r_{k,j} - r_{i,j}| + 1 \right).$$

On the event in Lemma 22, the event in Lemma 23, and the event $\mathcal{E}_0$, we have

$$|r_{\ell,j}^{(1)} - r_{i,j}| \leq \sqrt{\frac{2}{a}}$$

for all $X(\ell) \in \mathcal{C}_i^P$ and all $j \in [J]$.

Proof. Fix $i \in [I]$, fix $j \in [J]$, and fix $X(\ell) \in \mathcal{C}_i^P$. By definition,

$$r_{\ell,j}^{(1)} = \frac{\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h))\alpha_j(h)/n_j(h)}{\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h))}.$$

Hence,

$$|r_{\ell,j}^{(1)} - r_{i,j}| \leq \frac{\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h)) \left| \frac{\alpha_j(h)}{n_j(h)} - r_{i,j} \right|}{\sum_{h=1}^{H} p_\ell^{(1)}(\boldsymbol{\alpha}(h))}.$$

Split the numerator into the contribution from $\mathcal{H}_i$ and the contribution from the other clusters. On $\mathcal{E}_0$, we have

$$\left| \frac{\alpha_j(h)}{n_j(h)} - r_{i,j} \right| \leq \frac{1}{\sqrt{2a}}$$

for every $h \in \mathcal{H}_i$, and

$$\left| \frac{\alpha_j(h)}{n_j(h)} - r_{i,j} \right| \leq |r_{k,j} - r_{i,j}| + \frac{1}{\sqrt{2a}}$$

for every $h \in \mathcal{H}_k$ with $k \neq i$. Moreover, Lemma 24 implies $p_\ell^{(1)}(\boldsymbol{\alpha}(h)) \leq N^{-a\phi}$ for every $h \notin \mathcal{H}_i$. Since $X(\ell) \in \mathcal{C}_i^P$, we have $\sum_{h=1}^H p_\ell^{(1)}(\boldsymbol{\alpha}(h)) = H\rho_\ell^{(1)} \geq H\rho_T$. Therefore,

$$|r_{\ell,j}^{(1)} - r_{i,j}| \leq \frac{1}{\sqrt{2a}} + \frac{N^{-a\phi}\sum_{k\neq i}\sum_{h\in\mathcal{H}_k}\left(|r_{k,j} - r_{i,j}| + \frac{1}{\sqrt{2a}}\right)}{H\rho_T} \leq \frac{1}{\sqrt{2a}} + \frac{1 - \frac{H_i}{H}}{N^{a\phi}\rho_T}\left(\max_{1\leq k\leq I}|r_{k,j} - r_{i,j}| + \frac{1}{\sqrt{2a}}\right).$$

By Lemma 22, we have

$$1 - \frac{H_i}{H} \leq 1 - \frac{3}{4}\rho_i.$$

By the definition of $\underline{\rho}$ and the condition $\rho_T \geq \underline{\rho}$, the second term above is at most $1/\sqrt{2a}$. Hence, $|r_{\ell,j}^{(1)} - r_{i,j}| \leq \sqrt{\frac{2}{a}}$. This completes the proof. □

Lemma 27 (Pruning)**.** *If $|r_{\ell,j}^{(1)} - r_{i,j}| \leq \sqrt{\frac{2}{a}}$ for all $X(\ell) \in \mathcal{C}_i^P$ and all $j \in [J]$, then the pruning procedure identifies exactly one point from each $\mathcal{C}_i^P$.*

Proof. For each retained center, write $\boldsymbol{r}_\ell^{(1)} = (r_{\ell,1}^{(1)}, \ldots, r_{\ell,J}^{(1)})$. If $X(\ell), X(m) \in \mathcal{C}_i^P$, then

$$\|\boldsymbol{r}_\ell^{(1)} - \boldsymbol{r}_m^{(1)}\|_2^2 \leq \frac{8J}{a}.$$

If $X(\ell) \in \mathcal{C}_i^P$ and $X(m), X(m') \in \mathcal{C}_k^P$ with $i \neq k$, then

$$\|\boldsymbol{r}_m^{(1)} - \boldsymbol{r}_{m'}^{(1)}\|_2^2 \leq \frac{8J}{a},$$

and

$$\|\boldsymbol{r}_\ell^{(1)} - \boldsymbol{r}_m^{(1)}\|_2^2 \geq \sum_{j=1}^J \left(|r_{i,j} - r_{k,j}| - 2\sqrt{\frac{2}{a}}\right)^2.$$

It therefore suffices to show that

$$\sum_{j=1}^J \left(|r_{i,j} - r_{k,j}| - 2\sqrt{\frac{2}{a}}\right)^2 > \frac{8J}{a}$$

for every $i \neq k$. After expanding the square, this is implied by

$$\sum_{j=1}^J |r_{i,j} - r_{k,j}|^2 > 4\sqrt{\frac{2}{a}}\sum_{j=1}^J |r_{i,j} - r_{k,j}|.$$

Squaring both sides shows that the last inequality follows from

$$a > \frac{32\left(\sum_{j=1}^J |r_{i,j} - r_{k,j}|\right)^2}{\left(\sum_{j=1}^J |r_{i,j} - r_{k,j}|^2\right)^2}.$$

This is guaranteed by $a \geq \delta_4$. Hence there exists a constant Δ such that every within-cluster retained distance is strictly smaller than Δ and every between-cluster retained distance is strictly larger than Δ. After the traversal selects one point from some $\mathcal{C}_i^P$, every remaining point in that same set is within distance Δ of the selected set. At the same time, every point in any not-yet-represented $\mathcal{C}_k^P$ is farther than Δ from the selected set. Therefore, each subsequent Farthest First Traversal step must choose a point from a previously unrepresented cluster. Since there are exactly I nonempty sets $\mathcal{C}_1^P, \ldots, \mathcal{C}_I^P$, the traversal selects exactly one point from each of them. □

After pruning, relabel the selected centers so that retained center i comes from $\mathcal{C}_i^P$.

Lemma 28 (Second round soft assignments)**.** *If $\mathcal{E}_0$ holds and $|r_{i,j}^{(1)} - r_{i,j}| \le \sqrt{\frac{2}{a}}$ for all $i \in [I]$ and all $j \in [J]$, then $p_i^{(2)}(\boldsymbol{\alpha}(h)) \ge N^{a\phi} p_k^{(2)}(\boldsymbol{\alpha}(h))$ for all $h \in \mathcal{H}_i$ and all $i \ne k$.*

Proof. Fix $h \in \mathcal{H}_i$ and fix $k \ne i$. Since $a \ge \max\{\delta_1, \delta_2\}$, the center accuracy bound implies

$$\left|\log \frac{r_{i,j}^{(1)}}{r_{i,j}}\right| \le \sqrt{\frac{8}{a r_{i,j}^2}}, \qquad \left|\log \frac{1 - r_{i,j}^{(1)}}{1 - r_{i,j}}\right| \le \sqrt{\frac{8}{a(1 - r_{i,j})^2}},$$

and

$$\left|\log \frac{r_{k,j}^{(1)}}{r_{k,j}}\right| \le \sqrt{\frac{8}{a r_{k,j}^2}}, \qquad \left|\log \frac{1 - r_{k,j}^{(1)}}{1 - r_{k,j}}\right| \le \sqrt{\frac{8}{a(1 - r_{k,j})^2}}.$$

Because the second-round mixing weights are all equal to $1/I$, we have

$$\log \frac{p_i^{(2)}(\boldsymbol{\alpha}(h))}{p_k^{(2)}(\boldsymbol{\alpha}(h))} = \sum_{j=1}^{J} \alpha_j(h) \log \frac{r_{i,j}^{(1)}}{r_{k,j}^{(1)}} + \sum_{j=1}^{J} (n_j(h) - \alpha_j(h)) \log \frac{1 - r_{i,j}^{(1)}}{1 - r_{k,j}^{(1)}}.$$

Rearranging around the true parameters gives

$$\log \frac{p_i^{(2)}(\boldsymbol{\alpha}(h))}{p_k^{(2)}(\boldsymbol{\alpha}(h))} = \sum_{j=1}^{J} n_j(h) \Bigg[\mathrm{KL}(r_{i,j}, r_{k,j}) + \left(\frac{\alpha_j(h)}{n_j(h)} - r_{i,j}\right) \log \frac{r_{i,j}(1 - r_{k,j})}{r_{k,j}(1 - r_{i,j})} + \log \frac{1 - r_{i,j}^{(1)}}{1 - r_{i,j}} \frac{1 - r_{k,j}}{1 - r_{k,j}^{(1)}} + \frac{\alpha_j(h)}{n_j(h)} \left(\log \frac{r_{i,j}^{(1)}}{r_{i,j}} - \log \frac{r_{k,j}^{(1)}}{r_{k,j}} + \log \frac{1 - r_{k,j}^{(1)}}{1 - r_{k,j}} - \log \frac{1 - r_{i,j}^{(1)}}{1 - r_{i,j}} \right) \Bigg].$$

Using $\mathcal{E}_0$, the lower bound $n_j(h) \ge a \log N$, and the definitions of $A_{i,k}$, $B_{i,k}$, and $C_{i,k}$, we obtain

$$\log \frac{p_i^{(2)}(\boldsymbol{\alpha}(h))}{p_k^{(2)}(\boldsymbol{\alpha}(h))} \ge a \log N \left(2A_{i,k} - \frac{B_{i,k}}{\sqrt{a}} - \frac{C_{i,k}}{a} \right).$$

Since $a \ge \delta_3$, the right-hand side is at least $a\phi \log N$. Therefore,

$$\log \frac{p_i^{(2)}(\boldsymbol{\alpha}(h))}{p_k^{(2)}(\boldsymbol{\alpha}(h))} \ge a\phi \log N.$$

Exponentiating both sides completes the proof. □

Lemma 29 (Second round center estimates)**.** *If*

$$p_i^{(2)}(\boldsymbol{\alpha}(h)) \ge N^{a\phi} p_k^{(2)}(\boldsymbol{\alpha}(h))$$

for all $h \in \mathcal{H}_i$ and all $i \ne k$, then

$$\rho_i^{(2)} = \frac{H_i}{H}, \qquad r_{i,j}^{(2)} = \frac{1}{H_i} \sum_{h \in \mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)}.$$

Proof. Recall that

$$\widehat{z}(h) = \operatorname*{argmax}_{1\le i\le I} p_i^{(2)}(\boldsymbol{\alpha}(h)).$$

Fix $h \in \mathcal{H}_i$. The assumption implies

$$p_i^{(2)}(\boldsymbol{\alpha}(h)) > p_k^{(2)}(\boldsymbol{\alpha}(h))$$

for every $k \neq i$. Hence $\widehat{z}(h) = i$ for every $h \in \mathcal{H}_i$. Therefore,

$$\rho_i^{(2)} = \frac{1}{H}\sum_{h=1}^{H} \mathbb{1}\{\widehat{z}(h) = i\} = \frac{H_i}{H}.$$

The second-round reward update then gives

$$r_{i,j}^{(2)} = \frac{\sum_{h=1}^{H} \mathbb{1}\{\widehat{z}(h) = i\}\alpha_j(h)/n_j(h)}{\sum_{h=1}^{H} \mathbb{1}\{\widehat{z}(h) = i\}} = \frac{1}{H_i}\sum_{h\in\mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)}.$$

This completes the proof. □

Proof of Proposition 4. Let $\mathcal{E}_H$ be the event in Lemma 22. Let $\mathcal{E}_{\text{init}}$ be the event in Lemma 23. On the event $\mathcal{E}_H \cap \mathcal{E}_{\text{init}} \cap \mathcal{E}_0$, Lemma 25 shows that every $\mathcal{C}_i^P$ is nonempty. Lemma 26 then yields the first-round center accuracy bound. Lemma 27 shows that the pruning step retains exactly one center for each true type. Lemma 28 then implies that every customer is assigned to the correct cluster in the second round. Finally, Lemma 29 gives

$$\rho_i^{(2)} = \frac{H_i}{H}, \qquad r_{i,j}^{(2)} = \frac{1}{H_i}\sum_{h\in\mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)}.$$

By Lemma 22, Lemma 23, and the bound for $\mathcal{E}_0$, the intersection event $\mathcal{E}_H \cap \mathcal{E}_{\text{init}} \cap \mathcal{E}_0$ has probability at least $1 - \frac{2HJ}{N} - 2Ie^{-L\rho_{\min}/8} - 2Ie^{-H\rho_{\min}/48}$. This completes the proof. □

Lemma 30 (Concentration of empirical type frequencies)**.** *Assume that the customer types in the historical dataset are i.i.d. with $\mathbb{P}(z(h) = i) = \rho_i$ for all $h \in [H]$ and $i \in [I]$. Then, for every $\epsilon > 0$, we have*

$$\mathbb{P}\left(\max_{1\le i\le I}\left|\frac{H_i}{H} - \rho_i\right| \ge \epsilon\right) \le 2Ie^{-2H\epsilon^2}.$$

In particular, for each $i \in [I]$, we have

$$\frac{H_i}{H} \to \rho_i \quad \textit{almost surely as } H \to \infty.$$

Proof of Lemma 30. For each $i \in [I]$, we have

$$H_i = \sum_{h=1}^{H} \mathbb{1}\{z(h) = i\},$$

so $H_i \sim \text{Binomial}(H, \rho_i)$. Hoeffding's inequality yields

$$\mathbb{P}\left(\left|\frac{H_i}{H} - \rho_i\right| \ge \epsilon\right) \le 2e^{-2H\epsilon^2}.$$

Applying a union bound over $i \in [I]$ gives the displayed concentration bound. The almost sure convergence follows from the strong law of large numbers. □

Proof of Corollary 2. Let

$$\mathcal{E}_{\mathrm{pc}} := \left\{ \rho_i^{(2)} = \frac{H_i}{H} \text{ and } r_{i,j}^{(2)} = \frac{1}{H_i} \sum_{h \in \mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)} \text{ for all } i \in [I], j \in [J] \right\}.$$

By Proposition 4,

$$\mathbb{P}(\mathcal{E}_{\mathrm{pc}}^c) \le \frac{2HJ}{N} + 2Ie^{-L\rho_{\min}/8}.$$

For the population proportions, for each fixed i, the count H_i is Binomial(H, ρ_i). Hence Hoeffding's inequality gives

$$\mathbb{P}\left(\left| \frac{H_i}{H} - \rho_i \right| > \epsilon \right) \le 2e^{-2H\epsilon^2}.$$

Applying a union bound over $i \in [I]$ yields

$$\mathbb{P}\left(\max_{1 \le i \le I} \left| \frac{H_i}{H} - \rho_i \right| > \epsilon \right) \le 2Ie^{-2H\epsilon^2}.$$

Next define the event

$$\mathcal{E}_H := \left\{ \min_{1 \le i \le I} H_i \ge \frac{H\rho_{\min}}{2} \right\}.$$

Since $H_i \sim$ Binomial(H, ρ_i), a multiplicative Chernoff bound gives

$$\mathbb{P}(\mathcal{E}_H^c) \le \sum_{i=1}^{I} \mathbb{P}\left(H_i \le \frac{H\rho_i}{2} \right) \le Ie^{-H\rho_{\min}/8}.$$

For the reward matrix, fix $(i, j) \in [I] \times [J]$. Conditional on the true type assignments $\{z(h)\}_{h=1}^H$, the variables

$$Y_h^{(i,j)} := \frac{\alpha_j(h)}{n_j(h)}, \qquad h \in \mathcal{H}_i,$$

are independent, take values in $[0, 1]$, and satisfy

$$\mathbb{E}\left[Y_h^{(i,j)} \,\middle|\, \{z(h)\}_{h=1}^H \right] = r_{i,j}.$$

Therefore, conditional on $\{z(h)\}_{h=1}^H$, Hoeffding's inequality implies

$$\mathbb{P}\left(\left| \frac{1}{H_i} \sum_{h \in \mathcal{H}_i} Y_h^{(i,j)} - r_{i,j} \right| > \epsilon \,\middle|\, \{z(h)\}_{h=1}^H \right) \le 2e^{-2H_i\epsilon^2}.$$

On the event $\mathcal{E}_H$, we have $H_i \ge H\rho_{\min}/2$, so

$$2e^{-2H_i\epsilon^2} \le 2e^{-H\rho_{\min}\epsilon^2}.$$

Taking expectations and then applying a union bound over all (i, j) gives

$$\mathbb{P}\left(\mathcal{E}_H \cap \left\{ \max_{\substack{1 \le i \le I \\ 1 \le j \le J}} \left| \frac{1}{H_i} \sum_{h \in \mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)} - r_{i,j} \right| > \epsilon \right\} \right) \le 2IJe^{-H\rho_{\min}\epsilon^2}.$$

Finally, if either

$$\max_{1\le i\le I} |\rho_i^{(2)} - \rho_i| > \epsilon \qquad \text{or} \qquad \max_{\substack{1\le i\le I\\ 1\le j\le J}} |r_{i,j}^{(2)} - r_{i,j}| > \epsilon,$$

then at least one of the following must occur:

$$\mathcal{E}_{\text{pc}}^c, \qquad \left\{\max_{1\le i\le I} \left|\frac{H_i}{H} - \rho_i\right| > \epsilon\right\}, \qquad \mathcal{E}_H^c, \qquad \mathcal{E}_H \cap \left\{\max_{\substack{1\le i\le I\\ 1\le j\le J}} \left|\frac{1}{H_i}\sum_{h\in\mathcal{H}_i} \frac{\alpha_j(h)}{n_j(h)} - r_{i,j}\right| > \epsilon\right\}.$$

The stated probability bound now follows by a union bound. □