Hindsight Learning for MDPs with Exogenous Inputs

Sean R. Sinclair [1] **Felipe Frujeri** [2] **Ching-An Cheng** [2] **Luke Marshall** [2] **Hugo Barbalho** [2] **Jingling Li** [3]
Jennifer Neville [2] **Ishai Menache** [2] **Adith Swaminathan** [2]

Abstract

Many resource management problems require sequential decision-making under uncertainty, where the only uncertainty affecting the decision outcomes are exogenous variables outside the control of the decision-maker. We model these problems as Exo-MDPs (Markov Decision Processes with Exogenous Inputs) and design a class of data-efficient algorithms for them termed Hindsight Learning (HL). Our HL algorithms achieve data efficiency by leveraging a key insight: having samples of the exogenous variables, past decisions can be revisited in hindsight to infer counterfactual consequences that can accelerate policy improvements. We compare HL against classic baselines in the multi-secretary and airline revenue management problems. We also scale our algorithms to a business-critical cloud resource management problem – allocating Virtual Machines (VMs) to physical machines, and simulate their performance with real datasets from a large public cloud provider. We find that HL algorithms outperform domain-specific heuristics, as well as state-of-the-art reinforcement learning methods.

1. Introduction

Many aspects of our physical and digital infrastructure — like data centers, power grids, and supply chains — can become more adaptive and efficient through data-driven decision-making. For instance, in a world-wide cloud service, even 1% lower resource fragmentation can reduce energy use and save approximately $100M per year (Hadary et al., 2020). This type of improvement could be achieved by examining historical patterns of compute demands and using machine learning (ML) to allocate future demands more efficiently.

In this work, we make the key observation that in resource management applications the system is often partially known and the only uncertainty is due to *exogenous* variables like resource requests — that are (to a first-order approximation) *independent* of an agent's decisions (Powell, 2022). For example, a cloud operator deciding to place a virtual machine (VM) on a specific server rack does not directly affect future VM requests, but future demands can strongly affect the eventual quality of their allocation decisions (Hadary et al., 2020). We define these problems as *Exo-MDPs* (Markov Decision Processes with Exogenous Inputs), which are a subclass of *Input-Driven MDPs* (Mao et al., 2019b). In Input-Driven MDPs the minimal state describing the system dynamics decompose into (i) *exogenous* inputs, which evolve independently of the agent's actions, and (ii) *endogenous* factors that are impacted by the agent's actions and the exogenous inputs. Exo-MDPs make the additional assumption that the only unknowns are the distribution of future exogenous inputs (see §3).

ML has been applied in several resource management applications, which we will show are Exo-MDPs, and found to outperform domain-specific heuristics (Lykouris & Vassilvitskii, 2021; Kumar et al., 2018; Gollapudi & Panigrahi, 2019). The ML approaches often follow the Predict-Then-Optimize (PTO) paradigm (Elmachtoub & Grigas, 2022), using ML to forecast the future exogenous inputs (e.g., demands). However, when the future is highly stochastic, forecasting is challenging. For example, VM requests from real-world data-centers (Hadary et al., 2020) show long-term regularity of diurnal and weekly patterns but extreme short-term fluctuations (see Figure 4).

Reinforcement Learning (RL) is an alternative to PTO. RL directly optimizes decision quality (Chen et al., 2021; Fang et al., 2019; Mao et al., 2016) by replaying historical samples of the exogenous inputs through the known dynamics of endogenous factors to learn good policies (Madeka et al., 2022). RL methods applied to Exo-MDPs must, however, learn by trial-and-error that their actions can never affect the exogenous inputs. RL is thus sensitive to variance in the outcomes introduced by the exogenous inputs and requires

[1]School of Operations Research and Information Engineering, Cornell University [2]Microsoft Research, Redmond [3]Department of Computer Science, University of Maryland. Correspondence to: Sean Sinclair <srs429@cornell.edu>, Adith Swaminathan <adswamin@microsoft.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

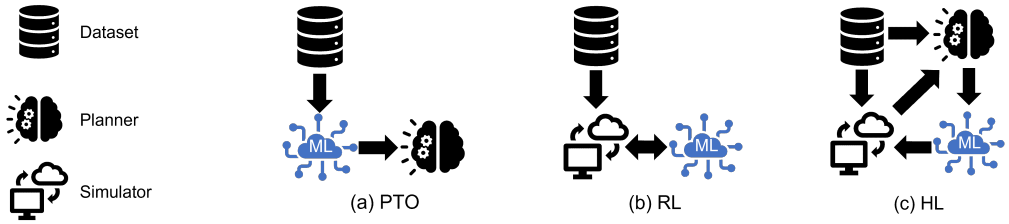


Figure 1: Conceptual view of different ML approaches to solving Exo-MDPs ((a) and (b) are detailed in Appendix E). Arrows in the figure indicate the flow of information from one component to the other. In (a) **Predict-Then-Optimize** (PTO) uses the dataset to train an ML forecasting model, uses the model to predict future exogenous inputs, and uses the forecasted inputs online for planning optimal actions. In (b) **RL** replays the dataset through the simulator to evaluate an ML policy's performance, and tunes policy parameters using the collected rewards as the training signal. In (c) our **Hindsight Learning** (HL) approach uses the dataset directly with the planner (top arrow) to identify hindsight-optimal values, and trains the ML policy using state trajectories from the simulator annotated with the hindsight optimal values.

more data to learn an optimal policy when the variance is high (Foster et al., 2021) (see §4).

Recent works have proposed to use *hindsight* control variates to reduce the variance of RL for input-driven MDPs (which include Exo-MDPs) (Mao et al., 2019b; Mesnard et al., 2021). They derive unbiased policy gradients by subtracting from the observed outcomes a function that depends additionally on hindsight information. However, we find that for many resource management scenarios, the variance reduction from hindsight control variates is not enough for data-efficient learning in practical regimes (see Table 2).

We argue that hindsight information can be more effectively used to improve learning, as it can largely reduce the variance of exogenous inputs at the cost of a small amount of asymptotic bias in many cases. Based on this insight, we develop a family of algorithms called Hindsight Learning (HL), which uses hindsight planners during learning to lessen the variance from exogenous inputs. A hindsight planner (Chong et al., 2000; Gopalan et al., 2010; Conforti et al., 2014) is an optimization algorithm that provides computationally tractable approximations to hindsight-optimal decisions, which is the optimal action specialized to a fixed sequence of future demands and thus is not affected by the exogenous variability. Therefore, by using hindsight planners during learning, HL can more efficiently identify decisions critical to future performance under high-variance exogenous inputs. Figure 1 contrasts the HL algorithm schematic with PTO and RL.

We theoretically characterize when HL succeeds using a novel quantity called *hindsight bias* (which arises due to the mismatch between truly optimal and hindsight-optimal decisions). Remarkably, we find that hindsight bias is small for many resource management problems, so HL can learn with extremely limited data for them. To prove this, we use recent advances in prophet inequalities (Dutting et al., 2020; Vera & Banerjee, 2021) with novel adaptations for the Exo-MDP setting. We empirically test HL in several domains: Multi-Secretary problems, Airline Revenue Management (ARM) benchmarks, and Virtual Machine (VM) allocation. We find that HL is significantly better than both domain heuristics and RL (with and without hindsight control variates), illustrating that HL indeed strikes a better bias-variance trade-off. Notably, our VM allocation experiments use real historical traces from a large public cloud provider, where HL is the only approach that consistently beats the currently used heuristics ($0.1\% - 5\%$ better). Recall that even a 1% better allocator can yield massive savings in practice (Hadary et al., 2020).

2. Related Work

Here we include a brief discussion of salient related work. Please see Appendix B for more details.

Recent studies have exploited causal structure in MDPs for better decision-making (Lattimore et al., 2016; Lu et al., 2022). Their causal graphs however do not capture Exo-MDPs (e.g., Figure 2). When the exogenous process is only partially observed, HL may additionally need causal RL techniques (Zhang et al., 2020b); this is left for future work.

Input-Driven MDPs have been specialized before with additional assumptions that either the rewards or the transitions factorize so that the exogenous process can be filtered out (Dietterich et al., 2018; Efroni et al., 2022). However, they are not suited for Exo-MDPs because filtering out the exogenous process yields demand-agnostic policies, which are highly sub-optimal for resource management problems.

Hindsight optimization has been previously attempted (Chong et al., 2000; Feng et al., 2021), and it is well known that these values are over-optimistic. Mercier & Van Hentenryck (2007) show that despite the over-optimism, the regret for hindsight optimization policies in several network scheduling and caching problems is

a constant. The Progressive Hedging (PH) algorithm in Rockafellar & Wets (1991) attempts to eliminate the over-optimism by iteratively refining the hindsight optimization solution by adding non-anticipative constraints. PH has weak guarantees for convergence in non-convex problems (like our settings) and is intractable for the large problem sizes that we consider. Information relaxation (Brown & Smith, 2022) extends PH to non-convex rewards and arbitrary action spaces and allows for imperfect penalization of the non-anticipative constraint violations. These schemes require hand-crafted penalties as well as tractable hindsight planning with those penalized objectives. Instead, Mercier & Van Hentenryck (2007) foregoes solving for the optimal policy of the MDP and instead produces a potentially sub-optimal non-anticipatory policy. We provide a tighter regret analysis for their surrogate policy (Lemma 13) and additionally describe an imitation learning algorithm to avoid unnecessary computation in large-scale problems.

3. Problem Setting and Definitions

3.1. MDPs with Exogenous Inputs (Exo-MDPs)

We consider a subclass of finite horizon Markov Decision Processes (MDPs). An MDP is defined by $(\mathcal{S}, \mathcal{A}, T, P, R, s_1)$ with horizon T, state space $\mathcal{S}$, action space $\mathcal{A}$, reward distribution R, state transition distribution P, and starting state s_1 (Puterman, 2014). An Exo-MDP further specializes an MDP by separating the process into *endogenous* and *endogenous* parts: a state $s \in \mathcal{S}$ factorizes into endogenous/system state $x \in \mathcal{X}$ and exogenous inputs $\boldsymbol{\xi} := (\xi_1, \dots, \xi_T) \in \Xi^T$ (namely, $\mathcal{S} := \mathcal{X} \times \Xi^T$). The state transition distribution P also factors into an endogenous part f and exogenous part $\mathcal{P}_\Xi$ as follows. At time t, the agent selects action $a_t \in \mathcal{A}$ based on the current state $s_t := (x_t, \boldsymbol{\xi}_{<t})$ where $\boldsymbol{\xi}_{<t} := (\xi_1, \dots, \xi_{t-1})$ is the observed exogenous inputs thus far, *and then* ξ_t is sampled from an unknown distribution $\mathcal{P}_\Xi(\cdot \mid \boldsymbol{\xi}_{<t})$, independently of the agent's action a_t. With ξ_t, the endogenous state evolves according to $x_{t+1} = f(s_t, a_t, \xi_t)$ and the reward earned is $r(s_t, a_t, \xi_t) \in [0, 1]$. Note ξ_t is only observed when the agent makes the decision at time $t+1$, *not* at time t. We restrict our attention to policies $\pi_t : \mathcal{X} \times \Xi^{t-1} \to \Delta(\mathcal{A})$ and let Π denote the policy class of the agent. The endogenous dynamics f and reward function r are assumed to be known to the agent[1]; the only unknown in the Exo-MDP is $\mathcal{P}_\Xi$. For notational convenience, we also assume that f and r are deterministic; all of our insights carry over to stochastic rewards and transitions. These assumptions of Exo-MDPs are well-motivated for resource management problems, and we list the state decomposition along with f and r for several examples in Appendix C.

[1]Thus, Exo-MDPs are a subclass of Input-Driven MDPs (Mao et al., 2019b) which more generally have unknown f, r.

3.2. Value Decomposition in Exo-MDPs

Since the only unknown in an Exo-MDP is $\mathcal{P}_\Xi$, a policy's performance can be written as expectations over $\mathcal{P}_\Xi$. This motivates the use of historical samples $\boldsymbol{\xi} \sim \mathcal{P}_\Xi$ to evaluate any policy and find optimal policies for the Exo-MDP.

For $\pi \in \Pi$, the *values* and *action-values* are defined as:

$$V_t^\pi(s) := \mathbb{E}_{\boldsymbol{\xi}_{\geq t}, \pi}[\sum_{\tau \geq t} r(s_\tau, a_\tau, \xi_\tau) \mid s_t = s],$$
$$Q_t^\pi(s, a) := \mathbb{E}_{\boldsymbol{\xi}_{\geq t}, \pi}[\sum_{\tau \geq t} r(s_\tau, a_\tau, \xi_\tau) \mid s_t = s, a_t = a],$$

where the expectation is taken over the randomness in π and the exogenous inputs $\boldsymbol{\xi}$. We denote $\pi^\star$ as the optimal policy, i.e. the policy that maximizes $V_t^\pi(s)$ in each state s, and denote $Q^\star, V^\star$ for $Q^{\pi^\star}, V^{\pi^\star}$ respectively. Our goal is to find a policy with near-optimal returns, $\arg\max_{\pi \in \Pi} \{V_1^\pi := V_1^\pi(s_1)\}$. Or equivalently, minimize $\textsc{Regret}(\pi)$ where $\textsc{Regret}(\pi) := V_1^\star - V_1^\pi$. For convenience we assume that $\pi^\star \in \Pi$. We introduce value functions for fixed $\boldsymbol{\xi} = \{\xi_1, \dots, \xi_T\}$ as

$$Q_t^\pi(s, a, \boldsymbol{\xi}_{\geq t}) := \mathbb{E}_\pi[\sum_{\tau \geq t} r(s_\tau, a_\tau, \xi_\tau) | s_t = s, a_t = a], \quad (1)$$
$$V_t^\pi(s, \boldsymbol{\xi}_{\geq t}) := \sum_a \pi(a|s) Q_t^\pi(s, a, \boldsymbol{\xi}_{\geq t}). \quad (2)$$

Note that the expectation is not over $\mathcal{P}_\Xi$ because $\boldsymbol{\xi}$ is fixed. The $\boldsymbol{\xi}$-specific values are related to policy values as follows.

Lemma 1. *For every $t \in [T], (s, a) \in \mathcal{S} \times \mathcal{A}$, policy $\pi \in \Pi$ we have that*

$$Q_t^\pi(s, a) = \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[Q_t^\pi(s, a, \boldsymbol{\xi}_{\geq t})], \quad (3)$$
$$V_t^\pi(s) = \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\pi(s, \boldsymbol{\xi}_{\geq t})]. \quad (4)$$

In particular $V_1^\pi = \mathbb{E}_{\boldsymbol{\xi}}[V_1^\pi(s_1, \boldsymbol{\xi})]$.

We relegate complete proofs to Appendix F. Since the transition dynamics f and reward function r are known and the unknown $\mathcal{P}_\Xi$ does not depend on the agent's actions, an immediate consequence is that *action exploration is not needed*. Given any policy π and exogenous trace $\boldsymbol{\xi}$ we can simulate with f and r to calculate its return in Equation 1.

Suppose we collected traces $\mathcal{D} = \{\boldsymbol{\xi}^1, \dots, \boldsymbol{\xi}^N\}$ where each trace $\boldsymbol{\xi}^n = \{\xi_1^n \dots, \xi_T^n\}$ is sampled independently from $\mathcal{P}_\Xi$. Finding a near-optimal policy using this historical dataset is known as the *offline RL* problem (Fujimoto et al., 2019; Liu et al., 2020; Rashidinejad et al., 2021; Cheng et al., 2022), but this is much simpler in Exo-MDPs. We do not face *support mismatch* wherein trajectories from a data-collection policy may not cover the scenarios that the learner policy would encounter. Here $\mathcal{D}$ (collected by a behavior policy) can be safely replayed to evaluate any learner policy. This fact also implies that model selection

and hyper-parameter tuning can be safely done using a held-out $\mathcal{D}$ akin to supervised learning. Our goal finally is to learn policies that generalize from $\mathcal{D}$ to the unknown $\mathcal{P}_\Xi$, which can be challenging because the exogenous inputs $\boldsymbol{\xi}$ introduce variance in a policy's return estimation.

3.3. Hindsight Planner

Exo-MDPs not only allow easy policy evaluation using a dataset of traces, but they also allow computing valuable hindsight information like the hindsight-optimal decisions for a trace $\boldsymbol{\xi}$. This hindsight information can be stable even when $\mathcal{P}_\Xi$ is highly stochastic. We now make a computational assumption for calculating hindsight-optimal decisions that will enable tractable algorithms for Exo-MDPs.

Assumption 1. *Given any trace $\boldsymbol{\xi}_{\geq t} = (\xi_t, \ldots, \xi_T)$ and state $s = (x_t, \boldsymbol{\xi}_{<t})$ we can tractably solve:*

$$\max_{a_t,\ldots,a_T} \sum_{\tau=t}^{T} r(s_\tau, a_\tau, \xi_\tau) \quad (5)$$
$$\text{s.t. } x_{\tau+1} = f(s_\tau, a_\tau, \xi_\tau), \text{ for } \tau = t, \ldots, T$$
$$s_\tau = (x_\tau, \boldsymbol{\xi}_{<\tau}), \text{ for } \tau = t, \ldots, T.$$

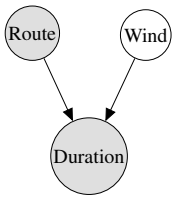
We denote the optimal objective value to this problem as $\textsc{Hindsight}(t, \boldsymbol{\xi}_{\geq t}, s)$.

The optimization community has developed computationally efficient implementations for the $\textsc{Hindsight}(t, \boldsymbol{\xi}_{\geq t}, s)$ oracle; with tight bounds on the optimal value even when Equation (5) is intractable. For example, online knapsack for a fixed input sequence can be solved in pseudo-polynomial time (Gopalan et al., 2010). In many instances, Equation 5 can be represented as an integer program and solved via heuristics (Conforti et al., 2014). Recently RLCO (RL for Combinatorial Optimization) has proved to be an effective heuristic for hindsight planning (Anthony et al., 2017; Fang et al., 2021). Note that Assumption 1 or RLCO cannot be used directly as a non-anticipatory policy for the Exo-MDP, since the whole sequence $\boldsymbol{\xi} \sim \mathcal{P}_\Xi$ is not observed upfront when making decisions. We discuss several examples of hindsight planners in Appendix D and assess the impact of approximate planning empirically in §7.3.2.

4. Using Hindsight In Exo-MDPs: An Example

In an Exo-MDP, the only uncertainty is due to unknown $\mathcal{P}_\Xi$. When $\mathcal{P}_\Xi$ introduces substantial variability in outcomes, the natural question is whether generic MDP algorithms can learn effectively? When $T = 1$, Exo-MDPs are isomorphic to a multi-armed bandit and so general bandit algorithms are optimal for the Exo-MDP. However, we will see in the next example with $T > 1$ that the answer is in general no.

Consider the sailing example in Figure 2 which is an Exo-MDP where the decision is to pick between one of two routes *prior to observing wind* with hopes of minimizing the trip duration. By direct calculation, $Q^\star(\text{route2}) - Q^\star(\text{route1}) = -48$. Hence, the optimal non-anticipative policy will always pick route2.



Wind ($\Pr(\xi)$)	route1	route2
East (0.49)	100	1
West (0.51)	50	51
$Q_1^\pi(\cdot, a)$	74.5	26.5

Figure 2: An Exo-MDP for sailing in uncertain winds: $\boldsymbol{\xi} = \{\text{Wind}\}, \mathcal{A} = \{\text{Route}\}, r = \{\text{Duration}\}$ and $\mathcal{X} = \emptyset$. First, the agent picks a *Route*. Then *Wind* conditions are observed during the trip and the agent receives a cost with respect to *Duration*. Values in the table denote average trip duration $r(a)$ (accounting for random fluctuations in wind).

RL: If a classic RL approach was applied to this problem, it would estimate the average duration for each route using observed samples, include exploration bonuses to account for uncertainty, and compare the averages to pick future routes. This requires many $\boldsymbol{\xi}$ samples because wind introduces large variance in the Q-estimates, and requires sufficient data to be collected across both the routes.

Hindsight Learning: In hindsight, we can instead use *all* observed samples (leveraging known f and r), with *no* exploration bonuses, and use paired comparisons to identify the optimal route. Variability due to wind means the routes' durations are typically positively correlated and thus a paired sample between routes will be more statistically efficient.

5. Hindsight Learning

We introduce Hindsight Learning (HL) to incorporate hindsight information in a principled manner, so as to reduce the exogenous variability and thereby speed-up learning. HL first uses a hindsight planner (Assumption 1) to derive a surrogate policy $\pi^\dagger$:

$$\pi_t^\dagger(s) \coloneqq \arg\max_{a \in \mathcal{A}} Q_t^\dagger(s, a), \quad (6)$$
$$Q_t^\dagger(s, a) \coloneqq \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[r(s, a, \xi_t) + \textsc{Hindsight}(t+1, \boldsymbol{\xi}_{>t}, f(s, a, \xi_t))], \quad (7)$$
$$V_t^\dagger(s) \coloneqq \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[\textsc{Hindsight}(t, \boldsymbol{\xi}_{\geq t}, s)]. \quad (8)$$

We define $Q_t^\dagger(s, a, \boldsymbol{\xi}_{\geq t})$ and $V_t^\dagger(s, \boldsymbol{\xi}_{\geq t})$ as the terms inside of the respective expectations. Note that $\pi^\dagger$ is a non-anticipatory policy, which considers expectation over future exogenous $\boldsymbol{\xi}_{\geq t}$ rather than being defined for a fixed trace. $\pi^\dagger$ is called "Bayes Selector" in the literature (Vera & Banerjee, 2021; Mercier & Van Hentenryck, 2007) and has been used for applications like bin packing and refugee resettle-

ment (Bansak & Paulson, 2022; Ahani et al., 2021; Banerjee & Freund, 2020). Intuitively $\pi^\dagger$ uses the returns accumulated by hindsight-optimal actions to score and rank good decisions, instead of mimicking the hindsight-optimal actions directly. However, it is always the case that $V_t^\dagger(s) \geq V^\star(s)$ and $Q_t^\dagger(s,a) \geq Q^\star(s,a)$ (Chong et al., 2000), and so $\pi^\dagger$ can be a sub-optimal surrogate for $\pi^\star$. In §6 we will bound its gap to $\pi^\star$ with a novel quantity called *hindsight bias* and discuss the implications for learning.

5.1. Imitating the "Bayes Selector"

Executing $\pi^\dagger$ requires frequent calls to the hindsight planner online to evaluate $Q_t^\dagger(S_t, a)$ on every observed state. Therefore running this *tabular* policy (i.e., considering every state separately) can be prohibitively costly when policy execution must also satisfy latency constraints (e.g., in VM allocation). Additionally, in resource allocation domains it is infeasible to enumerate all possible states. We describe a family of algorithms in Algorithm 1 that offloads the hindsight planner invocations to an offline training phase and distills $\pi^\dagger$ into a computationally feasible policy of neural networks that can extrapolate to unseen states.

Algorithm 1 Hindsight Learning

1: **Input:** An empty buffer $\mathcal{B}$, simulator for f and r, initial policy π, dataset $\mathcal{D}$, number of epochs K.
2: **for** $k = 1, 2, \ldots, K$ **do**
3: Sample a trace $\boldsymbol{\xi}$ from $\mathcal{D}$
4: Sample trajectory $\{s_1 \ldots s_T\}$ from π using the $f, r, \boldsymbol{\xi}$
5: Label sampled states from the trajectory with $\{Q_t^\dagger(s_t, a, \boldsymbol{\xi}_{\geq t}) : a \in \mathcal{A}, t \in [T]\}$
6: Aggregate the labeled data into the buffer $\mathcal{B}$
7: Optimize π on $\mathcal{B}$ either with Hindsight MAC (Equation 9) or Hindsight Q-Distillation (Equation 10)
8: **end for**

We use online imitation learning (IL) with $\pi^\dagger$ as the expert policy, specifically, the AggreVaTE algorithm (Ross & Bagnell, 2014) with expert $Q^\dagger(s,a)$ values from the hindsight planner. By interleaving the trajectory sampling and policy updates we avoid querying $Q^\dagger(s,a)$ (and hence solving hindsight planning problems) in uninformative states. We note that all of this interactive querying of the expert occurs during *offline* training, and hence the planning is not needed online during test time. Since we allow $\boldsymbol{\xi}$ to be arbitrarily correlated across t, in Algorithm 1 we sample an entire trace $\boldsymbol{\xi}^i$ from $\mathcal{D}$. However, if $\boldsymbol{\xi}$ is iid across time-steps t we can enhance step 3 by resampling a single ξ_t at each t.

Many popular RL algorithms (Konda & Tsitsiklis, 2000; Van Hasselt et al., 2016; Schulman et al., 2017) compute Q-values or advantages via Monte Carlo estimation. HL can be easily incorporated in them by replacing the Monte Carlo estimates (e.g., $Q^\pi(s,a)$) with Step 5 of Algorithm 1 (i.e., $Q^\dagger(s,a)$). We outline two such modifications below, one using a policy network and the other using a critic network. Since we can simulate $Q_t^\dagger(s, a, \boldsymbol{\xi}_{\geq t})$ for any action, we use the *common random numbers* insight (Ng & Jordan, 2000) for variance reduction and sum across all actions in both instantiations. This additional sum over actions trick is not critical, and is used in our experiments only because the action spaces are relatively small.

Hindsight MAC: We modify Mean Actor Critic (Allen et al., 2017) by incorporating $Q^\dagger$ into differentiable imitation learning (Sun et al., 2017). For the policy represented with a neural network π_θ, consider the loss function:

$$\ell(\pi_\theta) = \mathbb{E}_{\boldsymbol{\xi}}[\sum_{t=1}^{T} \mathbb{E}_{S_t \sim \Pr_t^\pi} \sum_{a \in \mathcal{A}} \pi_\theta(a \mid S_t) Q_t^\dagger(S_t, a, \boldsymbol{\xi}_{\geq t})]. \quad (9)$$

Hindsight Q-Distillation: We can represent $Q^\dagger$ values directly using a neural network critic Q^θ. The policy is defined implicitly w.r.t. Q^θ as $\pi_\theta = \arg\max_{a \in \mathcal{A}} Q^\theta(s,a)$. The loss function $\ell(\pi_\theta)$ to fit Q^θ is:

$$\ell(\pi_\theta) = \mathbb{E}_{\boldsymbol{\xi}}[\sum_{t=1}^{T} \mathbb{E}_{S_t \sim \Pr_t^\pi}[\sum_{a \in \mathcal{A}} (Q_t^\theta(S_t, a) - Q_t^\dagger(S_t, a, \boldsymbol{\xi}_{\geq t}))^2]]. \quad (10)$$

We optimize a sample approximation of Equation 9 or 10 using the dataset $\mathcal{D}$. The current policy defines the state sampling distribution, while $Q^\dagger$ gives the long-term reward signal for each action.

6. Theoretical Guarantees

Compared with RL, HL uses the known dynamics f and rewards r of an Exo-MDP, the hindsight planner of Assumption 1, and the dataset $\mathcal{D}$ to trade-off a small asymptotic bias (which we define in Equation (11)) for a large reduction in the variance from exogenous inputs. To prove this, we first characterize the regret of $\pi^\dagger$ in terms of a novel quantity, called the *hindsight bias*, and show that it is negligible in many resource management problems. Next we show that HL is sample efficient, and can imitate the $\pi^\dagger$ policy with faster optimization than RL. Finally, even if hindsight bias is large in an application, there are techniques to reduce it, including combinations of HL and RL in the future.

Definition 1. *The hindsight bias of $\pi^\star$ versus $\pi^\dagger$ at time t in state s is defined as*

$$\Delta_t^\dagger(s) \coloneqq Q_t^\dagger(s, \pi_t^\dagger(s)) - Q_t^\star(s, \pi_t^\dagger(s)) + Q_t^\star(s, \pi_t^\star(s)) - Q_t^\dagger(s, \pi_t^\star(s)). \quad (11)$$

Consider the over-estimation error of the hindsight planner $\Omega_t(s,a) \coloneqq Q_t^\dagger(s,a) - Q_t^\star(s,a)$ (referred to as *local loss* in Mercier & Van Hentenryck (2007)). Equation (11) subtracts the over-estimation of $\pi^\star(s)$ from the over-estimation of

$\pi^\dagger(s)$ and so the hindsight bias can be small even if the overestimation is large; as an extreme example consider the case when the argmax of $Q^\dagger$ and $Q^\star$ coincide (see Lemma 12). We show that $\Delta_t^\dagger(s)$ bounds the regret of $\pi^\dagger$.

Theorem 2. $\textsc{Regret}(\pi^\dagger) \leq \sum_{t=1}^T \mathbb{E}_{S_t \sim \Pr_t^{\pi^\dagger}}[\Delta_t^\dagger(S_t)]$, *where* $\Pr_t^{\pi^\dagger}$ *denotes the state distribution of* $\pi^\dagger$ *at step* t *induced by the exogenous process. In particular, if* $\Delta_t^\dagger(s) \leq \Delta$ *for some constant* Δ *then we have:* $\textsc{Regret}(\pi^\dagger) \leq \Delta \sum_{t=1}^T \mathbb{E}_{S_t \sim \Pr_t^{\pi^\dagger}}[\Pr(\pi_t^\dagger(S_t) \neq \pi^\star(S_t))]$.

Regret bounds of this form appear in the prophet inequality literature (Dutting et al., 2020; Vera et al., 2021) however against a much stronger benchmark of the hindsight planner, where the regret is defined as $V^\dagger(s_1) - V_1^\pi(s_1)$. Mercier & Van Hentenryck (2007) (Theorem 1) show that regret is bounded by the worst-case hindsight bias on the states visited by *any* decision policy. However, there are examples (see Lemma 13) where their bound is large and one could incorrectly conclude that hindsight optimization should not be applied. In contrast, Theorem 2 is tighter, requiring that the hindsight bias be small only on states visited by $\pi^\dagger$.

As corollaries of Theorem 2, the regret of $\pi^\dagger$ is *constant* for many resource management Exo-MDPs, such as stochastic online bin packing with i.i.d. arrivals. See Appendix G.1 (Lemma 18) for a formal statement of the result, and a discussion of related results from the literature.

Finally, we show that the performance of the best policy produced by Algorithm 1 will converge to that of $\pi^\dagger$ under standard assumptions for online imitation learning (Sun et al., 2017; Ross et al., 2011; Yan et al., 2021). We use the overline notation to denote quantities for an empirical MDP whose exogenous distribution $\mathcal{P}_\Xi$ is replaced with the empirical one $\overline{\mathcal{P}_\Xi} \sim \mathcal{D}$.

Theorem 3. *Let* $\overline{\pi}^\dagger$ *denote the hindsight planning surrogate policy for the empirical Exo-MDP w.r.t.* $\mathcal{D}$*. Assume* $\overline{\pi}^\dagger \in \Pi$ *and Algorithm 1 achieves no-regret in the optimization problem of Equation 9. Let* π *be the best policy from Algorithm 1. Then, for any* $\delta \in (0,1)$*, with probability* $1-\delta$*,*

$$\textsc{Regret}(\pi) \leq 2T\sqrt{\frac{2\log(2|\Pi|/\delta)}{N}} + \sum_{t=1}^T \mathbb{E}_{S_t \sim \overline{Pr}_t^{\overline{\pi}^\dagger}}[\bar{\Delta}_t^\dagger(S_t)] + o(1),$$

for $\bar{\Delta}_t^\dagger$ *the sample average of (11), and* $\overline{Pr}_t^{\overline{\pi}^\dagger}$ *is the state probability of* $\overline{\pi}^\dagger$ *in the empirical MDP.*

In Appendix E we also derive sample complexity results for PTO and RL when applied to Exo-MDPs. In PTO (see Theorem 6) the guarantees scale quadratically in T and depend on the complexity of $\mathcal{P}_\Xi$ (which in the worst case can scale by $|\Xi|^T$ if the ξ_t are strongly correlated across t). In contrast, Theorem 3 scales linearly in T and is only affected by the randomness over the induced $Q^\dagger$ values and not directly by the complexity of $\mathcal{P}_\Xi$. Unlike guarantees for RL (see Theorem 7), Theorem 3 is not asymptotically consistent due to the hindsight bias. However, RL methods have asymptotic consistency *only if* they converge to the optimal policy in the empirical MDP. This convergence is an idealized computation assumption that hides optimization issues when studying statistical guarantees and is incomparable to Assumption 1 for the hindsight planner (for which we show several examples in Appendix D).

Although hindsight bias is small for many practical Exo-MDPs, this is not universally true as we now show. Since hindsight bias bounds the regret of $\pi^\dagger$ (Theorem 2), Exo-MDPs with large regret must also have large hindsight bias.

Theorem 4. *There exists a set of Exo-MDPs such that* $\textsc{Regret}(\pi^\dagger) \geq \Omega(T)$.

Hence, for an arbitrary Exo-MDP the hindsight bias needs to be properly controlled to successfully leverage hindsight planning (Chong et al., 2000). The information relaxation literature (Brown & Smith, 2014; El Shar & Jiang, 2020) subtracts a carefully chosen baseline $b(s,a,\boldsymbol{\xi})$ in special cases of Equation (7); viewed through our results, this procedure essentially reduces the hindsight bias of the eventual $\pi^\dagger$. Building on this technique, we anticipate future works to design HL variants that are more robust to hindsight bias.

7. Experiments

We evaluate Hindsight Learning on three resource management domains with different characteristics (our code is available at https://github.com/seanrsinclair/hindsight-learning). First, **Multi-Secretary**, where the exogenous inputs are the arriving candidates' qualities and the hindsight bias is negligible (see Theorem 4.2 of Banerjee et al. (2020)). Next we consider **Airline Revenue Management** where the exogenous inputs are the current request's (resource demands, revenue) and the hindsight bias is small (see Lemma 18). Lastly, we consider **VM Allocation** where the exogenous inputs are VM requests and the hindsight bias is unknown. In Appendices C.2 and D we show explicit constructions of the Exo-MDP and hindsight planner for each domain. For the first two domains, we use traces drawn from benchmark distributions and evaluate $\pi^\dagger$ using Monte-Carlo rollouts. For the VM allocation domain we use real-world historical traces extracted from a large public cloud provider.

7.1. Multi-Secretary Problems

Multi-secretary is the generalization of the classic secretary problem (Buchbinder et al., 2009), where T candidates ar-

Table 1: Performance of heuristics, $\pi^\dagger$, RL and HL algorithms on multi-secretary and ARM problems benchmarked against the optimal policy. Values are V^π, the performance of the compared policy evaluated using the Bellman equations, and error bars computed via a standard normal approximation averaging over the randomly sampled dataset. Since these are tabular problems, HINDSIGHT MAC and HINDSIGHT Q-DISTILLATION are identical, so we report the performance of both as HINDSIGHT MAC. Relative performance compared against $V^{\pi^\star}$ is shown in parenthesis.

Multi-Secretary	$T=5$		$T=10$		$T=100$	
$\pi^\star$	2.22		5.09		49.9	
$\pi^\dagger$	2.21	(-0.5%)	4.95	(-2.7%)	49.85	(-0.2%)
Greedy	1.67	(-24.8%)	3.81	(-25.1%)	38.76	(-22.4%)
Tabular Q-learning	1.67 ± 0.0032	(-24.8%)	3.81 ± 0.0037	(-25.1%)	48.10 ± 0.027	(-3.7%)
Hindsight MAC	$\mathbf{2.17 \pm 0.0040}$	(-2.4%)	$\mathbf{4.98 \pm 0.0035}$	(-2.1%)	$\mathbf{48.65 \pm 0.022}$	(-2.6%)
ARM	$T=5$		$T=10$		$T=100$	
$\pi^\star$	1.89		3.72		39.03	
$\pi^\dagger$	1.88	(-0.3%)	3.61	(-2.9%)	37.27	(-4.5%)
Greedy	1.39	(-26.5%)	2.50	(-32.9%)	31.54	(-19.2%)
Tabular Q-learning	1.28 ± 0.015	(-32.2%)	2.75 ± 0.064	(-26.0%)	32.59 ± 0.25	(-16.5%)
Hindsight MAC	$\mathbf{1.81 \pm 0.032}$	(-4.0%)	$\mathbf{3.30 \pm 0.095}$	(-11.4%)	$\mathbf{33.84 \pm 0.37}$	(-13.3%)

rive sequentially but only B can be selected. An arriving candidate at time t has ability $r_t \in (0, 1]$ drawn i.i.d. from a finite set of K levels of expertise. At each round, if the decision-maker has remaining budget (i.e., has chosen less than B candidates thus far), they can *accept* a candidate and collect the reward r_t, or *reject* the candidate. The goal is to maximize the expected cumulative reward.

When T is large relative to N, we can expect historical traces to provide sufficient information about $\mathcal{P}_\Xi$. Recent results in Banerjee et al. (2020) use the "Bayes Selector" with a single exogenous trace to derive a policy with constant regret for a sufficiently large T. This suggests that the hindsight bias is negligible in this regime. Our experiment setup is identical to Banerjee et al. (2020) and is included in the supplementary material. We use $T = \{5, 10, 100\}, B = \frac{3}{5}T, K = 4$ and $N = 1$. The Greedy heuristic accepts the first B candidates regardless of their quality. ML methods use a single trace sampled from the non-stationary candidate arrival process, and use a policy that maps a 3-dim state (the rounds and budget remaining, and the current candidate ability) to an *accept* probability. For the hindsight planner, we use Equation 2 from Banerjee et al. (2020) which implements a linear program with $2K$ variables. The Bayes Selector $\pi^\dagger$ solves the LP with the historical trace in every possible state, and is only feasible for problems with small LPs and state spaces. We evaluate each policy using dynamic programming with the true arrivals distribution.

In Table 1 (Top) we see that the HL algorithm (Hindsight MAC) is competitive with the optimal policy (which depends on the unknown $\mathcal{P}_\Xi$ distribution) using just a *single exogenous trace*. RL (implemented via Tabular Q-learning) however is very sample inefficient; for small $T \leq 10$ it performs no better than the sub-optimal Greedy heuristic.

7.2. Airline Revenue Management

Airline Revenue Management (Littlewood, 1972) is a special case of the multi-dimensional Online Bin Packing (OBP) problem (OBP exhibits vanishing hindsight bias via Lemma 18). The agent has capacity B_k for K different resources. At each round, the decision-maker observes a request $A_t \in \mathbb{R}_+^K$ (the consumed capacity in each resource dimension), alongside a revenue f_t. The algorithm can either *accept* the request (obtaining revenue f_t and updating remaining capacity according to A_t), or *reject* it (note that partial acceptance is not allowed). The goal of the decision-maker is to maximize the expected revenue.

We use `ORSuite` (Archer et al., 2022) as an ARM simulator with fixed capacity, i.i.d. request types and job distribution, using a setting from Vera & Banerjee (2021) which shows large regret for existing heuristics. We vary T from 5 to 100, and compute $\pi^\star$ through dynamic programming. Both RL (Tabular Q-learning) and HL (Hindsight MAC) were trained on the same dataset of $N = 100$ traces.

In Table 1 (Bottom) we see that HL outperforms RL but is not as good as the Bayes Selector $\pi^\dagger$. Since the state space is much larger in ARM, HL has not sampled all the relevant states for imitating $\pi^\dagger$ and so its performance suffers. Moreover, as T increases the performance of RL again approaches HL, highlighting that HL strikes a better bias-variance trade-off to perform better with limited data.

7.3. VM Allocation

Arguably, our experiments thus far have been advantageous to HL because the hindsight bias is known to be small. We next examine HL on a large-scale allocation problem: allocating virtual machines (VMs) to physical servers. In contrast to previous experiments, in this problem, the bias can be arbitrary, and enumerating all states or solving the Bellman equations is infeasible. From an algorithmic perspective, the allocation problem is a multi-dimensional variant of OBP with stochastic (not i.i.d.) arrivals and departures (hence, the bound on hindsight bias does not apply). Due to problem scale we cannot compute $\pi^\star$ exactly with dynamic programming, so we instead benchmark a policy's performance with respect to a **BestFit** heuristic.

In the VM allocation problem we have K physical machines (PM), each with a fixed capacity limit for both CPU and memory. VM requests arrive over time, each with an associated CPU, memory requirement, and a lifetime (or duration); the lifetime is in principle unknown to the provider, but it can be predicted (Cortez et al., 2017). Accordingly, we study below two variants where lifetime information is either available (§7.3.1) or not (§7.3.2). The decision-maker must assign a feasible PM for the VM or reject the request (incurring a large penalty). A PM is considered active when one or more VMs are assigned to it. The objective is to minimize the total time that the machines remain active, normalized by the time horizon T (i.e., the average number of active PMs per unit time). This objective is critical for cloud efficiency; see Buchbinder et al. (2021) for a longer discussion.

7.3.1. Stylized environment

To gain insights into the problem domain, we consider first a stylized setting where the VMs arrive at discrete time steps (in practice, time is continuous, and VMs may arrive at any point in time); furthermore, the VM lifetime is perfectly predicted upon arrival. To carry out the experiments, we use the MARO simulator (Jiang et al., 2020) with $K = 80$ PMs and $T = 288$ (reflecting one day period discretized into time steps, each of which represents 5 minutes of actual time). MARO replays the VM requests in the Azure Public Dataset (Cortez et al., 2017)[2] as follows: all VM requests arriving within a time step (i.e., 5 minutes of actual time) are buffered and instead arrive simultaneously at the next discrete time step. The first half of the resulting trace is used for training and the remaining trace for testing. To evaluate any policy, we sampled 50 different one-day traces from the held-out portion and report the average value of the objective function. For the hindsight planner, we implemented the integer program of Appendix D.5 in Gurobi and solved its linear relaxation. We used a modified objective function (inverse packing density) which is linear in the decision variables for computational feasibility (see discussion in Appendix G).

[2]This dataset contains a uniform sample of VM requests received in a real data center over a one month period in 2019.

Table 2: Performance of heuristics, RL, and HL algorithms on VM allocation benchmarked against the Best Fit baseline. $\star$ indicate significant improvement and $\circ$ indicate significant decrease, over BestFit by Welch's t-test.

Algorithm	PMs Saved
Performance Upper Bound (Oracle)	$4.96^\star$
Best Fit	0.0
Bin Packing	-1.05°
DQN	-0.64
MAC	-0.51°
PPO	-0.50
PG with Hindsight Baseline (Mao et al., 2019b)	-0.057
HINDSIGHT MAC	$\mathbf{4.33}^\star$
HINDSIGHT Q-DISTILLATION	$3.71^\star$

We compare four allocation approaches. (1) **Heuristics**: We consider several heuristics that have been widely used for different bin packing problems (round robin, first fit, load balance, etc.). We report here the results for the best performing heuristic *BestFit*, which has been widely applied in practice (Panigrahy et al., 2011), in particular for VM allocation (Hadary et al., 2020); in a nutshell BestFit chooses the machine which leaves less amount of unused resources (see (Panigrahy et al., 2011) for details). (2) **RL**: We benchmark several popular RL algorithms including Double-DQN (Van Hasselt et al., 2016), MAC (Allen et al., 2017) and PPO (Schulman et al., 2017). (3) **Hindsight approaches:** We test Hindsight MAC (Equation 9) and Hindsight Q-Distillation (Equation 10). In addition, we test Mao et al. (2019b) which uses hindsight-aware control variates to reduce the variance of policy gradient (PG) methods. (4) **Oracle:** We report the $\textsc{Hindsight}(1, \boldsymbol{\xi}, s_1)$ (objective of the relaxed IP) evaluated on the test traces, and use the experiment outcome as a performance upper bound.

All the ML methods use a 4-layer neural net to map features describing a PM and the VM request to a score. In Appendix G, we detail the network design, state features and the hyper-parameter ranges we used. Table 2 reports the *PMs Saved* which is the regret for the objective function relative to *BestFit*, averaged across the evaluation traces. We created realistic starting state distributions by executing the *BestFit* heuristic for a random duration (greater than one day). Error bars are computed by (i) training each algorithm over 20 random seeds (neural network parameters and the offline dataset) and (ii) evaluating each algorithm on 50

one-day traces sampled from the hold-out set. We then compared its performance to Best Fit on each evaluation trace with a paired t-test of value $p = 0.05$. We observe that HL outperforms all the heuristics and RL methods, requiring 4 fewer PMs on average (or a 5% improvement in relative terms, since $K = 80$).

7.3.2. Real-World Resource Allocation

We now consider a more realistic setting where VM arrivals are in continuous time and the allocation agent has no information about VM lifetimes. In real-world settings, the scale of clusters can be much larger than the one considered in §7.3.1, see Hadary et al. (2020); scaling ML algorithms to larger inventory sizes is an ongoing research direction. Furthermore, each VM arrival or departure is modeled as a step in the Exo-MDP, resulting in much larger time horizon T (order of 100k). Our total trace period was 88 days, and we used the exact methodology as in §7.3.1 to obtain the training and test datasets. Due to the large scale, even the linear relaxation of the integer program was not tractable. Consequently, we carefully designed a *hindsight heuristic* (Algorithm 3) to derive $\textsc{Hindsight}(t, \boldsymbol{\xi}, s)$. The heuristic prioritizes VMs according to both their size and lifetime (see Appendix G.6.3).

Table 3: Average number of PMs saved by RL and HL policies across 5 clusters, calculated over 44 days and benchmarked against the production **BestFit** heuristic. $\star$ indicate significant improvement and $\circ$ indicate a significant decrease, over BestFit by Welch's t-test.

Cluster	A	B	C	D	E
RL	-0.14	-0.35	-0.27°	$1.34^{\star}$	-0.37
HL	$\mathbf{3.20}^{\star}$	$\mathbf{1.35}$	$\mathbf{1.13}^{\star}$	$\mathbf{2.27}^{\star}$	$\mathbf{0.02}$

We adapt Hindsight MAC (HL) and compare it with MAC (Allen et al., 2017) (RL), where both used the same network architecture, which embeds VM-specific and PM-specific features using a 6-layer GNN. The resulting architecture is rich enough to represent the **BestFit** heuristic, but can also express more flexible policies. The Bayes Selector $\pi^{\dagger}$ is infeasible to run within the latency requirements for VM allocation, and so is not compared.

Table 3 summarizes the results over five different clusters. Unlike §7.3.1 where we sampled many 1-day periods from the test trace, the demands on the real clusters were non-stationary throughout the test period. Hence we report results on the entire 44-day test trace. We trained each algorithm over 3 random seeds and evaluated 5 rollouts to capture the variation in the cluster state at the start of the evaluation trace. Unlike the other experiments, we cannot account for the randomness in exogenous samples because we only have one evaluation trace for each cluster. Error metrics are computed with a paired t-test of value $p = 0.05$.

We observe that RL exhibits unreliable performance: in fact, it is sometimes worse than **BestFit**, intuitively this can happen because it overfits to the request patterns seen during training. In contrast, HL always improved over **BestFit**, with relative improvements of $0.1\% - 1.6\%$ over RL and $0.1\% - 0.7\%$ over BestFit (note cluster sizes are much larger here than §7.3.1). As noted earlier, any percent-point (or even fractions of a percent) improvement implies millions of dollars in savings. The relative gains obtained here are more modest than in the stylized setting due to a combination of reasons. First, intuitively, every packing "mistake" is more costly in a smaller cluster, meaning that algorithms have more room to shine in smaller-scale problems. Second, using a heuristic for hindsight learning is inherently sub-optimal. Lastly, we have not used *any* information about VM lifetime; an interesting direction for future work is incorporating lifetime predictions to HL.

8. Conclusion

In this paper, we introduced Hindsight Learning (HL) as a family of algorithms that solve a subclass of MDPs with exogenous inputs, termed Exo-MDPs. Exo-MDPs capture a variety of important resource management problems, such as VM allocation. We show that the HL algorithms outperform both heuristics and RL methods. One direction for future work is to blend RL with HL using reward shaping (Cheng et al., 2021) for solving Exo-MDPs with large hindsight bias. Intuitively, combining pessimistic value estimates from RL with optimistic estimates from HL can provide finer-grained control for trading-off hindsight bias and variance from exogenous inputs. Another direction is designing hindsight learning algorithms in "nearly Exo-MDP" environments where the action can have a limited impact on the exogenous variables, such as using recent results from Liu et al. (2021).

Acknowledgements

We thank Janardhan Kulkarni, Beibin Li, Connor Lawless, Siddhartha Banerjee, and Christina Yu for inspiring discussions. We thank Dhivya Eswaran, Tara Safavi, and Tobias Schnabel for reviewing early drafts. Part of this work was done while Sean Sinclair and Jingling Li were research interns at Microsoft Research, and while Sean Sinclair was a visitor at Simons Institute for the semester on Data-Driven Decision Processes program. We gratefully acknowledge funding from the National Science Foundation under grants ECCS-1847393, DMS-1839346, CCF-1948256, CNS-195599, and CNS-1955997, the Air Force Office of Scientific Research under grant FA9550-23-1-0068, and the Army Research Laboratory under grants W911NF-19-1-0217 and W911NF-17-1-0094.

References

Abbeel, P. and Ng, A. Y. Exploration and apprenticeship learning in reinforcement learning. In *ICML*, pp. 1–8, 2005.

Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. Reinforcement Learning: Theory and Algorithms. Technical report, University of Washington, 2019. Available from https://rltheorybook.github.io.

Agrawal, S. and Jia, R. Learning in structured mdps with convex cost functions: Improved regret bounds for inventory management. *Operations Research*, 70(3):1646–1664, 2022.

Ahani, N., Gölz, P., Procaccia, A. D., Teytelboym, A., and Trapp, A. C. Dynamic placement in refugee resettlement. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pp. 5–5, 2021.

Allen, C., Asadi, K., Roderick, M., Mohamed, A.-r., Konidaris, G., and Littman, M. Mean actor critic. *arXiv preprint arXiv:1709.00503*, 2017.

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Archer, C., Banerjee, S., Cortez, M., Rucker, C., Sinclair, S. R., Solberg, M., Xie, Q., and Lee Yu, C. Orsuite: Benchmarking suite for sequential operations models. *ACM SIGMETRICS Performance Evaluation Review*, 49 (2), 2022.

Balseiro, S. R. and Brown, D. B. Approximations to stochastic dynamic programs via information relaxation duality. *Operations Research*, 67(2), 2019.

Banerjee, S. and Freund, D. Uniform loss algorithms for online stochastic decision-making with applications to bin packing. In *SIGMETRICS*, 2020.

Banerjee, S., Gurvich, I., and Vera, A. Constant regret in online allocation: On the sufficiency of a single historical trace, 2020.

Bansak, K. and Paulson, E. Outcome-driven dynamic refugee assignment with allocation balancing. In *EC*, 2022.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. In *ICLR*, 2017.

Bertsimas, D. and Tsitsiklis, J. N. *Introduction to linear optimization*, volume 6. Athena, 1997.

Borgs, C., Chayes, J. T., Lovász, L., Sós, V. T., and Vesztergombi, K. Convergent sequences of dense graphs I: Subgraph frequencies, metric properties and testing. *Advances in Mathematics*, 219(6), 2008.

Brown, D. B. and Haugh, M. B. Information relaxation bounds for infinite horizon markov decision processes. *Operations Research*, 65(5), 2017.

Brown, D. B. and Smith, J. E. Information relaxations, duality, and convex stochastic dynamic programs. *Operations Research*, 62(6), 2014.

Brown, D. B. and Smith, J. E. Information relaxations and duality in stochastic dynamic programs:: A review and tutorial. *Foundations and Trends in Optimization*, 5(3), 2022.

Bubeck, S. and Cesa-Bianchi, N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1), 2012.

Buchbinder, N., Jain, K., and Singh, M. Secretary problems and incentives via linear programming. *ACM SIGecom Exchanges*, 8(2), 2009.

Buchbinder, N., Fairstein, Y., Mellou, K., Menache, I., and Naor, J. Online virtual machine allocation with lifetime and load predictions. *ACM SIGMETRICS Performance Evaluation Review*, 49(1), 2021.

Chen, D., Chen, K., Li, Z., Chu, T., Yao, R., Qiu, F., and Lin, K. PowerNet: Multi-agent deep reinforcement learning for scalable powergrid control. *IEEE Transactions on Power Systems*, 37(2), 2021.

Cheng, C.-A., Kolobov, A., and Swaminathan, A. Heuristic-guided reinforcement learning. In *NeurIPS*, 2021.

Cheng, C.-A., Xie, T., Jiang, N., and Agarwal, A. Adversarially trained actor critic for offline reinforcement learning. In *International Conference on Machine Learning*, pp. 3852–3878. PMLR, 2022.

Chitnis, R. and Lozano-Pérez, T. Learning compact models for planning with exogenous processes. In *Conference on Robot Learning*, pp. 813–822. PMLR, 2020.

Chong, E. K., Givan, R. L., and Chang, H. S. A framework for simulation-based network control via hindsight optimization. In *CDC*, volume 2, 2000.

Conforti, M., Cornuéjols, G., and Zambelli, G. *Integer programming*, volume 271. Springer, 2014.

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., and Bianchini, R. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *SOSP*, 2017.

Dai, J. G. and Gluzman, M. Queueing network controls via deep reinforcement learning. *Stochastic Systems*, 2021.

Dietterich, T., Trimponias, G., and Chen, Z. Discovering and Removing Exogenous State Variables and Rewards for Reinforcement Learning. In *ICML*, 2018.

Domingues, O. D., Ménard, P., Kaufmann, E., and Valko, M. Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited. In *ALT*, 2021.

Dutting, P., Feldman, M., Kesselheim, T., and Lucier, B. Prophet inequalities made easy: Stochastic optimization by pricing nonstochastic inputs. *SIAM Journal on Computing*, 49(3), 2020.

Efroni, Y., Foster, D. J., Misra, D., Krishnamurthy, A., and Langford, J. Sample-efficient reinforcement learning in the presence of exogenous information. In *Conference on Learning Theory*, pp. 5062–5127. PMLR, 2022.

El Shar, I. and Jiang, D. Lookahead-bounded q-learning. In *ICML*, 2020.

Elmachtoub, A. N. and Grigas, P. Smart "predict, then optimize". *Management Science*, 68(1), 2022.

Fang, J., Ellis, M., Li, B., Liu, S., Hosseinkashi, Y., Revow, M., Sadovnikov, A., Liu, Z., Cheng, P., Ashok, S., Zhao, D., Cutler, R., Lu, Y., and Gehrke, J. Reinforcement learning for bandwidth estimation and congestion control in real-time communications. *arXiv preprint arXiv:1912.02222*, 2019.

Fang, Y., Ren, K., Liu, W., Zhou, D., Zhang, W., Bian, J., Yu, Y., and Liu, T.-Y. Universal trading for order execution with oracle policy distillation. In *AAAI*, 2021.

Feinberg, E. A. Optimality conditions for inventory control. In *Optimization Challenges in Complex, Networked and Risky Systems*. INFORMS, 2016.

Feng, J., Gluzman, M., and Dai, J. G. Scalable deep reinforcement learning for ride-hailing. In *American Control Conference*, 2021.

Foster, D. J., Kakade, S. M., Qian, J., and Rakhlin, A. The statistical complexity of interactive decision making. *arXiv preprint arXiv:2112.13487*, 2021.

Freund, D. and Banerjee, S. Good prophets know when the end is near. SSRN Scholarly Paper ID 3479189, Social Science Research Network, 2019.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Ghaderi, J., Zhong, Y., and Srikant, R. Asymptotic optimality of bestfit for stochastic bin packing. *ACM SIGMETRICS Performance Evaluation Review*, 42(2), 2014.

Goldberg, D. A., Reiman, M. I., and Wang, Q. A Survey of Recent Progress in the Asymptotic Analysis of Inventory Systems. *Production and Operations Management*, 30 (6), 2021.

Gollapudi, S. and Panigrahi, D. Online algorithms for rent-or-buy with expert advice. In *ICML*, 2019.

Gopalan, P., Klivans, A., and Meka, R. Polynomial-time approximation schemes for knapsack and related counting problems using branching programs. *arXiv preprint arXiv:1008.3187*, 2010.

Gupta, V. and Radovanovic, A. Online stochastic bin packing. *arXiv preprint arXiv:1211.2687*, 2012.

Hadary, O., Marshall, L., Menache, I., Pan, A., Greeff, E. E., Dion, D., Dorminey, S., Joshi, S., Chen, Y., Russinovich, M., and Moscibroda, T. Protean:VM allocation service at scale. In *OSDI*, 2020.

Harsha, P., Jagmohan, A., Kalagnanam, J., Quanz, B., and Singhvi, D. Math programming based reinforcement learning for multi-echelon inventory management. In *NeurIPS Deep RL Workshop*, 2021.

Hart, P. E., Stork, D. G., and Duda, R. O. *Pattern classification*. John Wiley & Sons, 2000.

Hu, H., Zhang, X., Yan, X., Wang, L., and Xu, Y. Solving a new 3d bin packing problem with deep reinforcement learning method. *arXiv preprint arXiv:1708.05930*, 2017.

Hubbs, C. D., Perez, H. D., Sarwar, O., Sahinidis, N. V., Grossmann, I. E., and Wassick, J. M. Or-gym: A reinforcement learning library for operations research problems. *arXiv preprint arXiv:2008.06319*, 2020.

Jiang, A., Zhang, J., Yu, P., Huang, L., Qiu, Y., Wang, J., Shi, W., Li, K., Wang, Z., Zhang, C., Sun, T., Chen, M., Yu, K., Wei, X., Li, M., Shang, N., Meng, Q., Li, S., Bian, J., Cheng, B., and Liu, T.-Y. Maro: A multi-agent resource optimization platform, 2020. URL https://github.com/microsoft/maro.

Kallus, N. and Zhou, A. Stateful offline contextual policy evaluation and learning. In *AISTATS*, 2022.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *NeurIPS*, 2000.

Kumar, R., Purohit, M., and Svitkina, Z. Improving online algorithms via ml predictions. In *NeurIPS*, 2018.

Langford, J. and Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. In *NeurIPS*, 2007.

Lattimore, F., Lattimore, T., and Reid, M. D. Causal bandits: learning good interventions via causal inference. In *NeurIPS*, 2016.

Li, Y., Tang, X., and Cai, W. Dynamic bin packing for on-demand cloud resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 27(1), 2015.

Littlewood, K. Forecasting and control of passenger bookings. In *Airline Group International Federation of Operational Research Societies*, 1972.

Liu, V., Wright, J., and White, M. Exploiting action impact regularity and partially known models for offline reinforcement learning. *arXiv preprint arXiv:2111.08066*, 2021.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch off-policy reinforcement learning without great exploration. *Advances in neural information processing systems*, 33:1264–1274, 2020.

Lovász, L. and Szegedy, B. Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B*, 96(6), 2006.

Lu, Y., Meisami, A., and Tewari, A. Efficient reinforcement learning with prior causal knowledge. In *Conference on Causal Learning and Reasoning*, 2022.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. *Journal of the ACM (JACM)*, 68 (4):1–25, 2021.

Madeka, D., Torkkola, K., Eisenach, C., Foster, D., and Luo, A. Deep inventory management. *arXiv preprint arXiv:2210.03137*, 2022.

Mao, H., Alizadeh, M., Menache, I., and Kandula, S. Resource management with deep reinforcement learning. In *HotNets*, 2016.

Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., and Alizadeh, M. Learning scheduling algorithms for data processing clusters. In *ACM Special Interest Group on Data Communication*, 2019a.

Mao, H., Venkatakrishnan, S. B., Schwarzkopf, M., and Alizadeh, M. Variance reduction for reinforcement learning in input-driven environments. In *ICLR*, 2019b.

Mercier, L. and Van Hentenryck, P. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. In *IJCAI*, pp. 1979–1984, 2007.

Mesnard, T., Weber, T., Viola, F., Thakoor, S., Saade, A., Harutyunyan, A., Dabney, W., Stepleton, T. S., Heess, N., Guez, A., Moulines, E., Hutter, M., Buesing, L., and Munos, R. Counterfactual credit assignment in model-free reinforcement learning. In *ICML*, 2021.

Ng, A. Y. and Jordan, M. Pegasus: a policy search method for large mdps and pomdps. In *UAI*, 2000.

Panigrahy, R., Talwar, K., Uyeda, L., and Wieder, U. Heuristics for vector bin packing. Technical report, Microsoft Research, 2011. Available from https://www.microsoft.com/en-us/research/wp-content/uploads/2011/01/VBPackingESA11.pdf.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Perboli, G., Tadei, R., and Baldi, M. M. The stochastic generalized bin packing problem. *Discrete Applied Mathematics*, 160(7-8), 2012.

Powell, W. *Reinforcement Learning and Stochastic Optimization: A unified framework for sequential decisions*. John Wiley & Sons, 2022.

Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., and Russell, S. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716, 2021.

Rockafellar, R. T. and Wets, R. J.-B. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

Ross, S. and Bagnell, J. A. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. A tutorial on thompson sampling. *Foundations and Trends in Machine Learning*, 11(1), 2018.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sheng, J., Cai, S., Cui, H., Li, W., Hua, Y., Jin, B., Zhou, W., Hu, Y., Zhu, L., Peng, Q., Zha, H., and Wang, X. VMAgent: Scheduling simulator for reinforcement learning. *arXiv preprint arXiv:2112.04785*, 2021.

Sheng, J., Hu, Y., Zhou, W., Zhu, L., Jin, B., Wang, J., and Wang, X. Learning to schedule multi-NUMA virtual machines via reinforcement learning. *Pattern Recognition*, 121, 2022.

Slivkins, A. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2), 2019.

Song, J., Lanka, R., Zhao, A., Bhatnagar, A., Yue, Y., and Ono, M. Learning to search via retrospective imitation. *arXiv preprint arXiv:1804.00846*, 2018.

Stolyar, A. L. An infinite server system with general packing constraints. *Operations Research*, 61(5), 2013.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B., and Bagnell, J. A. Deeply aggrevated: Differentiable imitation learning for sequential prediction. In *ICML*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tang, Y., Agrawal, S., and Faenza, Y. Reinforcement learning for integer programming: Learning to cut. In *ICML*, 2020.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Venuto, D., Lau, E., Precup, D., and Nachum, O. Policy gradients incorporating the future. In *ICLR*, 2022.

Vera, A. and Banerjee, S. The bayesian prophet: A low-regret framework for online decision making. *Management Science*, 67(3), 2021.

Vera, A., Banerjee, S., and Gurvich, I. Online allocation and pricing: Constant regret via bellman inequalities. *Operations Research*, 69(3), 2021.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *NeurIPS*, 2015.

Warrington, A., Lavington, J. W., Scibior, A., Schmidt, M., and Wood, F. Robust asymmetric learning in POMDPs. In *ICML*, 2021.

Weitzman, M. L. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, 1979.

Xin, L. and Goldberg, D. A. Distributionally robust inventory control when demand is a martingale. *Mathematics of Operations Research*, 47(3), 2022.

Yan, X., Boots, B., and Cheng, C.-A. Explaining fast improvement in online imitation learning. In *UAI*, 2021.

Zhang, H., Geng, X., and Ma, H. Learning-driven interference-aware workload parallelization for streaming applications in heterogeneous cluster. *IEEE Transactions on Parallel and Distributed Systems*, 32(1), 2020a.

Zhang, J., Kumor, D., and Bareinboim, E. Causal imitation learning with unobserved confounders. In *NeurIPS*, 2020b.

A. Table of Notation

Symbol	Definition
	Problem Setting Specification
$\mathcal{S}, \mathcal{A}, T, s_1, R, P$	MDP primitives: state and action space, time horizon, starting state, reward function and transition probabilities
$\mathcal{X}$	Endogenous space for the system
Ξ	Exogenous input space
$\mathcal{P}_\Xi$	Distribution over exogenous inputs
$f(s,a,\xi), r(s,a,\xi)$	Underlying deterministic transition and reward as function of exogenous input
s_t	MDP state space primitive, $(x_t, \boldsymbol{\xi}_{<t})$ for shorthand
s_t, a_t, ξ_t	State, action, and exogenous input for time step t
$\boldsymbol{\xi}$	An exogenous input trace $(\xi_1, \ldots, \xi_T)$
$\boldsymbol{\xi}_{\geq t}$	Component of an exogenous input trace $(\xi_t, \ldots, \xi_T)$
Π	Set of all admissible policies
$Q_t^\pi(s,a), V_t^\pi(s)$	Q-Function and value function for policy π at time step t
$\pi^\star, Q_t^\star(s,a), V_t^\star(s)$	Optimal policy and the Q and value function for the optimal policy
$\Pr_t^\pi$	State-visitation distribution for policy π at time step t
$\mathcal{D}$	Dataset containing N traces of exogenous inputs $\{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^N\}$
$\textsc{Hindsight}(t, s, \boldsymbol{\xi}_{\geq t}, f)$	Hindsight optimal cumulative reward r starting in state s at time t with exogenous inputs dictated by $\boldsymbol{\xi}_{\geq t}$ and dynamics f (see Equation (5))
$Q_t^\pi(s,a,\boldsymbol{\xi}_{\geq t}), V_t^\pi(s,a,\boldsymbol{\xi}_{\geq t})$	Q and V functions for policy π starting from s where exogenous inputs are given by $\boldsymbol{\xi}_{\geq t}$ (see Lemma 1)
$\overline{\mathbb{E}}[\cdot]$	Empirical expectation taken where $\boldsymbol{\xi}$ is sampled uniformly from $\mathcal{D}$
$\overline{\pi}^\star$	Policy obtained by ERM, i.e. solving $\arg\max_{\pi\in\Pi} \overline{\mathbb{E}}[V_1^\pi(s_1, \boldsymbol{\xi})]$.
$\overline{\mathcal{P}_\Xi}, \overline{Q}_t, \overline{V}_t, \overline{\pi}$	Estimated exogenous distribution using $\mathcal{D}$, $Q_t^\star$, $V_t^\star$ estimates using the estimated distribution, and the resulting policy
	Hindsight Planner
$Q_t^\dagger(s,a,\boldsymbol{\xi}_{\geq t})$	$r(s,a,\xi_t) + \textsc{Hindsight}(t+1, \boldsymbol{\xi}_{>t}, f(s,a,\xi_t))$
$V_t^\dagger(s,\boldsymbol{\xi}_{\geq t})$	$\textsc{Hindsight}(t, \boldsymbol{\xi}_{\geq t}, s)$
$Q_t^\dagger(s,a), V_t^\dagger(s,a)$	Expectations of $Q_t^\dagger(s,a,\boldsymbol{\xi}_{\geq t})$ and $V_t^\dagger(s,\boldsymbol{\xi}_{\geq t})$ over $\boldsymbol{\xi}_{\geq t}$
$\pi^\dagger$	Greedy policy with respect to $Q^\dagger$
$\Delta_t^\dagger(s)$	Hindsight bias for state s at time step t (see Equation (11))
Δ	Absolute bound on $\Delta_t^\dagger(s)$
$\overline{\mathcal{P}_\Xi}$	Empirical distribution over $\boldsymbol{\xi}$ from $\mathcal{D}$
$\overline{\pi^\dagger}$	Greedy policy with respect to $\overline{Q^\dagger}$ where true expectation over $\mathcal{P}_\Xi$ replaced with $\overline{\mathcal{P}_\Xi}$
$\bar{\Delta}_t^\dagger(s)$	Value of $\Delta_t^\dagger(s)$ where expectation over $\mathcal{P}_\Xi$ replaced with $\overline{\mathcal{P}_\Xi}$
$\overline{\Pr}_t^\pi$	State visitation distribution of π at time step t with exogenous dynamics $\overline{\mathcal{P}_\Xi}$

Table 4: List of common notation

B. Detailed Related Work

There is an extensive literature on reinforcement learning and its connection to tasks in operations management; below, we highlight the work which is closest to ours, but for more extensive references, see Sutton & Barto (2018); Agarwal et al. (2019); Powell (2022) for RL, and Bubeck & Cesa-Bianchi (2012); Slivkins (2019) for background on multi-armed bandits.

Information Relaxation for MDP Control: Information relaxation as an approach for calculating performance bounds on the optimal $Q^\star$ function has been developed recently using rich connections to convex duality (Vera & Banerjee, 2021; Brown & Smith, 2022; Balseiro & Brown, 2019; Brown & Haugh, 2017; Kallus & Zhou, 2022; Mercier & Van Hentenryck, 2007). As discussed in the main paper, for general problems, using hindsight planning oracles as in Assumption 1 creates estimates for the $Q^\star$ value which are overly optimistic of their true value. These differences can be rectified by introducing a control variate, coined *information penalties*, to penalize the planner's access to future information that a truly non-anticipatory

policy would not have. The goal is to construct penalties which ensure that the estimates of $Q^\star$ are truly consistent for the underlying value. This work has been developed explicitly in the context of infinite horizon MDPs (Brown & Haugh, 2017) where constructions are given for penalty functions as a function of the future randomness of the $\boldsymbol{\xi}$ process. Moreover, concrete algorithmic implementations using hindsight planners and information penalties has been developed in the tabular setting with no finite sample guarantees (El Shar & Jiang, 2020). Constructing these penalties in practice using suitable functions for arbitrary $\boldsymbol{\xi}$ is unknown. Our work differs by foregoing consistency of the estimates to instead focus on showing that in problem domains of interest, the policy which is greedy with respect to the hindsight planner is indeed consistent.

Behaviour Cloning: One approach for using hindsight information is behavior cloning. This will compute the hindsight-optimal actions for every $\boldsymbol{\xi} \sim \mathcal{D}$, and learn to imitate these actions using a feasible non-anticipatory policy (Fang et al., 2021). This is an instance of the *probability matching* principle which is widely used in Thompson sampling (Russo et al., 2018; Hart et al., 2000). Unfortunately, this *value-agnostic* approach is uncontrollably biased. Consider the example in §4. Since the winds in $\mathcal{D}$ will be west 51% of the time, the hindsight-optimal distribution (marginalizing over wind) is $\Pr(\text{route1}) = 0.51; \Pr(\text{route2}) = 0.49$. A non-anticipatory learner policy trained via behavior cloning will converge either to this distribution (if learning a stochastic policy) or its mode (using a deterministic policy). Both these policies are very sub-optimal compared to the optimal policy.

Policy Based Methods with Control Variates: Recent work has developed black box tools to modify policy gradient algorithms with control variates that depend on the exogenous trace. Recall that a typical policy-based algorithm uses either on-policy data (or off-policy with re-weighted importance sampling strategies), and estimates the gradient in the return via

$$\nabla V^{\pi_\theta} = \mathbb{E}_{S\sim \Pr_t^{\pi_\theta}, A\sim\pi_\theta}[\nabla \log \pi_\theta(A \mid S)\hat{Q}^{\pi_\theta}(S,A)].$$

From here, most methods subtract an appropriate baseline (commonly taken to be an estimate of the value function) as a form of Rao-Blackwellization to reduce the variance of the estimator while incurring no additional bias. In particular, for any function $b : \mathcal{S} \to \mathbb{R}$ we can instead take

$$\nabla V^{\pi_\theta} = \mathbb{E}_{S\sim \Pr_t^{\pi_\theta}, A\sim\pi_\theta}[\nabla \log \pi_\theta(A \mid S)(Q_{\pi_\theta}(S,A) - b(S))]$$

while remaining unbiased. However, due to the exogenous input structure on the MDP any function $b : \mathcal{X} \times \Xi^T \to \mathbb{R}$ also results in an unbiased gradient. Through this, the existing literature has taken different approaches for constructing these *input driven baselines*. In Mao et al. (2019b) they consider directly using a baseline of the form $b(x, \boldsymbol{\xi})$. As an architecture to learn a network representation of this baseline the authors propose either using a multi-value network or meta learning. In Mesnard et al. (2021) they consider using future conditional value estimates for the policy gradient baseline. In particular, they use Ψ_t as a new statistic to calculate new information from the rest of the trajectory and learn value functions which are conditioned on the additional hindsight information contained in Ψ_t. They provide a family of estimators, but do not specify which form of Ψ_t to use in generating an algorithm.

Recurrent Neural Network Policy Design: A related line of work modifies black box policy gradient methods by using a recurrent neural network (RNN) explicitly in policy design. In Venuto et al. (2022) they augment the state space to include $\boldsymbol{\xi}$ while simultaneously limiting information flow in the neural network to ensure that the algorithm is not overly relying on this privileged information. This approach, named *policy gradients incorporating the future*, is easy to implement as it just augments the network using an LSTM and adds a new loss term to account for the information bottleneck.

Learning to Search: The Exo-MDP model is closely related to the learning-to-search model. Expert iteration (Anthony et al., 2017) separates planning and generalization when learning to search, and provides an alternative approach to implement the HINDSIGHT oracle. Retrospective imitation (Song et al., 2018) faces a similar challenge as us: a given $\boldsymbol{\xi}$ defines a fixed search space and we seek search policies that generalize across $\mathcal{P}_\Xi(\boldsymbol{\xi})$. However, retrospective imitation reduces to realizable imitation problems because the learner witnesses $\boldsymbol{\xi}$ beforehand whereas in Exo-MDPs, $\boldsymbol{\xi}_{\geq t}$ is privileged information and imitating HINDSIGHT is typically unrealizable. Asymmetric Imitation Learning (Warrington et al., 2021) studies problems when imitating an expert with privileged information but essentially use RNN policies to ameliorate unrealizability.

RL for OR: In our work we primarily consider simulations on dynamic Virtual Machine (VM) scheduling. On the theoretical side, variants of greedy algorithms have been usually proposed to solve the dynamic VM scheduling problems with competitive ratio analysis. In Stolyar (2013) they assume the VM creation requests can be modeled as a Poisson process with lifetimes as an exponential distribution and show that the greedy algorithm achieves the asymptotically optimal policy. In Li et al. (2015) they develop a hybrid FIRSTFIT algorithm with an improvement on the competitive ratio. On the more practical side using deep reinforcement learning techniques, in Mao et al. (2016) they develop a DeepRM system which can

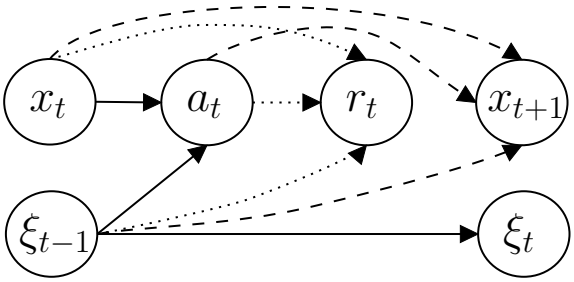


Figure 3: Causal diagram for an Exo-MDP where $k = 1$. Here the dotted line indicates the influence of (x_t, a_t, ξ_t) on the immediate reward r_t via $r(x_t, a_t, \xi_t)$ and the dashed line on the transition evolution as $x_{t+1} = f(s_t, a_t, \xi_t)$. The key facet to notice is the lack of influence on the ξ process from the current endogenous state x_t and action a_t.

pack tasks with multiple resource demands via a policy gradient method. They also built a job scheduler named DECIMA by modifying actor critic algorithms with input driven baselines (Mao et al., 2019a). In Zhang et al. (2020a) they solved the heterogeneous scheduling problem with deep Q learning. Lastly, in Sheng et al. (2022) they developed SchedRL, a modification of Deep Q Learning with reward shaping to develop a VM scheduling policy. All of these algorithms modify existing RL algorithms and show empirical gains on variations of the VM scheduling problem. Our work differs from two perspectives: 1) we consider using hindsight planning explicitly during training time, 2) our algorithms can be applied to any Exo-MDP problems.

We also note that existing deep reinforcement learning has also been applied in other systems applications (without exploiting their exo-MDP structure) including ride-sharing systems (Feng et al., 2021), stochastic queueing networks (Dai & Gluzman, 2021), power grid systems (Chen et al., 2021), jitter buffers (Fang et al., 2019), and inventory control (Harsha et al., 2021).

RL for Combinatorial Optimization: A crucial assumption underpinning our algorithmic framework is the implementation of hindsight planners as in Assumption 1. Our framework is well motivated for problems where hindsight planning is efficient, building on the existing optimization literature on solving planning problems (Conforti et al., 2014; Bertsimas & Tsitsiklis, 1997). However, in general these problems as a function of a fixed exogenous input trace can be written as combinatorial optimization problems. While we consider using hindsight planners for RL problems, a dual lens is using machine learning techniques for combinatorial optimization, as has been explored in recent years (Vinyals et al., 2015; Bello et al., 2017; Chitnis & Lozano-Pérez, 2020). In particular, in Vinyals et al. (2015) they designed a new network architecture and trained using supervised learning for traveling salesman problems. Similarly in Hu et al. (2017) they solve variants of online bin packing problems using policy gradient algorithms. In Tang et al. (2020) they design novel heuristic branch and bound algorithms using machine learning for integer programming optimization.

Exo-MDPs: Exo-MDPs, as highlighted in §3 were described in Powell (2022). They characterize sequential decision making problems as an evolution of *information*, *decision*, *information* sequence represented mathematically as the sequence $(s_1, a_1, \xi_1, s_2, a_2, \xi_2, \ldots, s_T)$. Here, the state variable s_t is written explicitly to capture the information available to the decision maker to make a decision a_t, followed by the information we learn after making a decision, i.e. the exogenous information ξ_t. Similar models have been outlined in (Mao et al., 2019b; Dietterich et al., 2018; Efroni et al., 2022).

C. MDPs with Exogenous Inputs

In this section we further discuss the definition of Exo-MDPs and its relations to contextual bandits and MDPs and highlight some examples in the operations management literature.

C.1. Generality of MDPs with Exogenous Inputs

As highlighted in §3 we consider the finite horizon reinforcement learning setting where an agent is interacting with a Markov Decision Process (MDP) (Puterman, 2014). The underlying MDP is given by a five-tuple $(\mathcal{S}, \mathcal{A}, T, p, R, s_1)$ where T is the horizon, $(\mathcal{S}, \mathcal{A})$ denotes the set of states and actions, R is the reward distribution, p the distribution governing the transitions of the system, and s_1 is the given starting state.

Definition 2. *In an **MDP with Exogenous Inputs** (Exo-MDP) we let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_T)$ be a trace of exogenous inputs with*

each ξ_t supported on the set Ξ. We assume that $\boldsymbol{\xi}$ is sampled according to an unknown distribution $\mathcal{P}_\Xi$. The agent has access to an endogenous or system state $x \in \mathcal{X}$. With this, the dynamics and rewards of the Markov decision process evolve where at time t, the agent selects their action $a_t \in \mathcal{A}$ based solely on $s_t = (x_t, \boldsymbol{\xi}_{<t})$. After, the endogenous state evolves according to $x_{t+1} = f(s_t, a_t, \xi_t)$, and the reward earned is $r(s_t, a_t, \xi_t)$, and ξ_t is observed. We assume that f and r are known by the principal in advance.

Note that this imposes that the state space for the underlying MDP can be written as $\mathcal{S} = \mathcal{X} \times \Xi^T$ where the first component corresponds to the endogenous state and the second to the exogenous input trace observed so far. We use the shorthand s_t to refer to $(x_t, \boldsymbol{\xi}_{<t})$.

As written, the distribution $\mathcal{P}_\Xi$ can be arbitrarily correlated across time. We can relax this setting to assume that $\boldsymbol{\xi}$ evolves according to a k-Markov chain. More formally, that at each step t, $\xi_t \mid (\xi_{t-k}, \xi_{t-k+1}, \ldots, \xi_{t-1})$ is conditionally independent of $(\xi_1, \ldots, \xi_{t-k-1})$. This allows the state space to be represented as $\mathcal{S} = \mathcal{X} \times \Xi^k$. Lastly, the dataset $\mathcal{D}$ contains a series of N traces sampled independently according to $\mathcal{P}_\Xi$ as $\mathcal{D} = \{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^N\}$ where each $\boldsymbol{\xi}^i = \{\xi_1^i, \ldots, \xi_T^i\}$.

For more intuition, consider the model under various values of k:

- **Case** $k = T$: Here we assume that $\boldsymbol{\xi}$ is an arbitrarily correlated process and $\mathcal{S} = \mathcal{X} \times \Xi^T$ so that $s_t = (x_t, \boldsymbol{\xi}_{<t})$. An example of this is VM allocation, where exogenous VM requests are highly correlated across time (Hadary et al., 2020).
- **Case** $k = 1$: Here we assume that $\boldsymbol{\xi}$ process evolves according to a 1-Markov chain. The state space factorizes as $\mathcal{X} \times \Xi$ where $\mathcal{X}$ is the endogenous space and Ξ is the exogenous space. The current state is $s_t = (x_t, \xi_{t-1})$, and the state updates to $(f(s_t, a_t, \xi_t), \xi_t)$ where ξ_t is drawn from the conditional distribution given ξ_{t-1}. A representation of the causal diagram under this setting is in Figure 3. An example of this is bin-packing, where it is typically assumed that jobs arrive according to a Markov chain.
- **Case** $k = 0$: Here we have that $\mathcal{S} = \mathcal{X}$. After taking an action a_t based solely on x_t we transition to $x_{t+1} = f(x_t, a_t, \xi_t)$ with ξ_t sampled independently from an unknown distribution $\mathcal{P}_\Xi^T$. The previous variable ξ_t can be either observed or unobserved. An example of this is inventory control or newsvendor models, where the demand is typically assumed to be i.i.d. across periods.

Relation between Contextual Bandits, MDPs, and Exo-MDPs We first notice that Exo-MDPs are a bridge Between Contextual Bandit and MDPs. When $\mathcal{X}$ is empty or a singleton, an Exo-MDP describes several variants of the *contextual bandit* introduced in Langford & Zhang (2007). They can be solved efficiently independent of the horizon (Foster et al., 2021), unlike MDPs. If $|\Xi| \leq 1$, an Exo-MDP is simply an MDP whose complexity scales with $|\mathcal{X}|$. When both $|\mathcal{X}| > 1$ and $|\Xi| > 1$ the complexity of learning an Exo-MDP is not known in general, to the best of our knowledge. When the exogenous inputs are iid, an Exo-MDP is equivalent to an MDP with state space $\mathcal{X}$ much smaller than $\mathcal{S}$.

Difference in Dataset Assumptions We next focus briefly on the case when $k = 0$, showing that the key difference between Exo-MDPs and the typical MDP models is the assumptions on the historical dataset provided. The typical assumptions in an MDP involve that $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ where the underlying distribution P is unknown. This can be written equivalently as $s_{t+1} = f(s_t, a_t, \xi_t)$ where ξ_t is sampled uniformly in $[0, 1]$ and the underlying function f is unknown. As such, typically in an MDP we consider:

- Unknown structure of the dynamics and rewards (i.e. unknown f and r)
- Known distribution on the underlying exogenous inputs where each ξ_t is uniform over $[0, 1]$
- Access to a logged dataset of (s_t, a_t, r_t, s_{t+1}) pairs

In Exo-MDPs we instead assume:

- Known structure of the dynamics and rewards (i.e. known f and r)
- Unknown distribution on the exogenous inputs $\mathcal{P}_\Xi$
- Access to a dataset of exogenous traces $\xi_1, \ldots, \xi_T$

These types of assumptions (where the *form* of the randomness is known but the true underlying distribution is unknown) is common in the graphon literature (Borgs et al., 2008; Lovász & Szegedy, 2006). In the following lemma we show that these two models are equivalent, in that any MDP can be written as an MDP with exogenous inputs and $k = 0$ using the uniform random number trick. However, the *assumptions are not equivalent* since in Exo-MDPs we assume access to a dataset of historical exogenous traces rather than trajectories under a fixed behavior policy.

Lemma 5. *Any MDP of the form $(\mathcal{S}, \mathcal{A}, T, p, R, s_1)$ where the distribution on p and R are unknown has an equivalent Exo-MDP form with $k = 0$, and vice-versa.*

Proof. Without loss of generality we will assume that both Ξ and $\mathcal{S}$ are either discrete or one dimensional (where higher dimensions follow via the same chain of reasoning).

Exo-MDP $\to$ MDP: Suppose that $s_{t+1} = f(s_t, a_t, \xi_t)$ where ξ_t is sampled from $\mathcal{P}_\Xi$ and f is known.

We can write this of the form where f is unknown and $\mathcal{P}_\Xi$ is known by setting $\widetilde{\xi}_t \sim U[0, 1]$ and $\widetilde{f}(s_t, a_t, \widetilde{\xi}_t) = f(s_t, a_t, \mathcal{P}_\Xi^{-1}(\widetilde{\xi}_t))$. Here the form of $\widetilde{f}$ is unknown as we cannot evaluate $\mathcal{P}_\Xi^{-1}$, but the distribution on the underlying randomness $\widetilde{\xi}$ is known.

MDP $\to$ Exo-MDP: Suppose that $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ where the distribution is unknown. We can write this as $s_{t+1} = f(s_t, a_t, \xi_t)$ with a known f and unknown distribution $\mathcal{P}_\Xi$ as follows.

First set $\Xi = \Delta(\mathcal{S})^{\mathcal{S}\times\mathcal{A}} \times [0, 1]$. Given any $\xi \in \Xi$ we define the transition kernel $f(s_t, a_t, \xi)$ as follows:

- Set $\tilde{p} \in \Delta(\mathcal{S})$ to be the component of ξ indexed via s_t, a_t
- Letting z be the last component of ξ, set $s_{t+1} = \tilde{p}^{-1}(z)$.

The distribution over Ξ is then defined as an indicator variable over the first $\mathcal{S} \times \mathcal{A}$ components indicating the true unknown distribution p, and the last component over $[0, 1]$ being Uniform$[0, 1]$. □

C.2. Examples of Exo-MDPs

We now give several examples of Exo-MDPs alongside with their exogenous decomposition of the transition distribution. We also highlight the underlying Markovian assumption on the exogenous inputs $\boldsymbol{\xi}$.

C.2.1. Inventory Control with Lead Times and Lost Sales ($k = 0$)

This models a single product stochastic inventory control problem with lost sales and lead times (Agrawal & Jia, 2022; Goldberg et al., 2021; Xin & Goldberg, 2022; Feinberg, 2016). In the beginning of every time step t, the inventory manager observes the current inventory level $\texttt{Inv}_t$ and L previous unfulfilled orders in the pipeline, denoted $o_L, \ldots, o_1$ for a product. L denotes the lead time or delay in the number of time steps between placing an order and receiving it. The next inventory is obtained as follows. First, o_1 arrives and the on-hand inventory rises to $I_t = \texttt{Inv}_t + o_1$. Then, an exogenous demand ξ is drawn independently from the unknown demand distribution $\mathcal{P}_\Xi$. The cost to the inventory manager is

$$h(I_t - \xi)^+ + p(\xi - I_t)^+$$

where h is the holding cost for remaining inventory and p is the lost sales penalty. The on-hand inventory then finishes at level $(I_t - \xi)^+$.

This can be formulated as an Exo-MDP by letting $\mathcal{X} = [n]^{L+1}$ denote the current inventory level and previous orders, $\Xi = [n]$ as the exogenous demand, and $\mathcal{A} = [n]$ for the amount to order where n is some maximum order amount. The reward function is highlighted above, and the state transition updates as $x' = f(x_t, a_t, \xi)$ where $\texttt{Inv}_{t+1} = (\texttt{Inv}_t + o_1 - \xi)^+$, $o_k = o_{k-1}$ for all $1 < k < L$ and $o_L = a$. This model can also be expanded to include multiple suppliers with different lead times.

C.2.2. Online Stochastic Bin Packing ($k = 1$)

Here we consider a typical online stochastic bin packing model (Gupta & Radovanovic, 2012; Ghaderi et al., 2014; Perboli et al., 2012) where a principal has access to an infinite supply of bins with maximum bin size B. Items u_t arrive over a

sequence of rounds $t = 1, \ldots, T$ where each $u_t \in [B]$ denotes the item size. At every time step, the principal decides on a bin to allocate the item to, either allocating it to a previously opened bin or creating a new bin. The goal is to allocate all of the items using the smallest number of bins.

This can be modeled in the framework as follows. Here we let $\mathcal{X} = \mathbb{R}^B$, $\Xi = [B]$ and $\mathcal{A} = [B]$. Each vector $x \in \mathcal{X}$ has components $x_1, \ldots, x_B$ as the current number of bins opened with current utilization of one up to B, with Ξ corresponding to the current item arrival's size. Hence the state space is $\mathcal{S} = \mathcal{X} \times \Xi$ where $s_t = (x_t, \xi_{t-1})$ corresponds to the current bin capacity and current item arrival. Actions $a \in \mathcal{A}$ correspond to either 0, for opening up a new bin, or $1, \ldots, B$ to be adding the current item to an existing bin with current utilization one up to B. The reward is:

$$r(x_t, \xi_{t-1}, a, \xi_t) = \begin{cases} -1 & a = 0 \\ 0 & a > 0, s_a > 0, \text{ and } a + \xi_{t-1} \leq B \\ -100 & \text{otherwise} \end{cases}$$

where -1 corresponds to the cost for opening a new bin, and the condition on zero reward verifies whether or not there is currently an open bin at level a and the action is feasible (i.e. allocating the current item to the bin at size a is smaller than the maximum bin capacity).

The transition distribution is updated similarly. Let ξ_t be drawn from the conditional distribution given ξ_{t-1}. If $a = 0$ then $x' = x$ except $x'_{\xi_{t-1}}$ is incremented by one (for opening up a new bin at the level of the size of the current item). If $a > 0$ and the action is feasible (i.e. $s_a > 0$ and $a + \xi_{t-1} \leq B$) then $x' = x$ with $x'_{a+\xi_{t-1}}$ incremented by one and x'_a decreased by one.

We note again that this model can be extended to include different reward functions, multiple dimensions of item arrivals, and departures, similar to the Virtual Machine allocation scenario.

C.2.3. Online Secretary ($k = 1$)

Multi-secretary is the generalization of the classic secretary problem (Buchbinder et al., 2009), where T candidates arrive sequentially but only B can be selected. Over time periods, a candidate arrives with ability $r_t \in (0, 1]$ drawn i.i.d. from a finite set of K levels of expertise. At each round, if the decision-maker has remaining budget (i.e., has chosen less than B candidates thus far), they can *accept* a candidate and collect the reward r_t, or *reject* the candidate. The goal is to maximize the expected cumulative reward.

This can be modeled as an Exo-MDP as follows. Here we let $\mathcal{X} = [B]$, $\Xi = [K]$, and $\mathcal{A} = \{0, 1\}$. The endogenous space $\mathcal{X}$ corresponds to the number of remaining candidates that can be accepted. The exogenous space Ξ corresponds to the ability level of the next time period's candidate. Lastly, actions $a \in \mathcal{A}$ correspond to either accepting or rejecting the current candidate. Hence the state space is $\mathcal{S} = \mathcal{X} \times \Xi$ where $s_t = (x_t, \xi_{t-1})$ corresponds to the number of accepted candidates thus far and the skill of the current candidate. The reward is:

$$r(x_t, \xi_{t-1}, a, \xi_t) = \begin{cases} \xi_{t-1} & a = 1 \text{ and } x_t > 0 \\ 0 & \text{otherwise} \end{cases}$$

The transition distribution is updated similarly by accounting for whether a candidate was accepted. Indeed we have:

$$f(x_t, \xi_{t-1}, a, \xi_t) = \begin{cases} x_{t-1} & a = 1 \text{ and } x_t > 0 \\ 0 & \text{otherwise} \end{cases}$$

C.2.4. Airline Revenue Management ($k = 0$)

Airline Revenue Management (Littlewood, 1972) is a special case of the multi-dimensional Online Bin Packing (OBP) problem, but we reiterate its model here for completeness. There are a set of $K \in \mathbb{N}$ resources, and each resource i has a maximum capacity B_i. Customers are segmented into $M \in \mathbb{N}$ types. Customers of type $j \in [M]$ request $A_j \in \mathbb{R}_+^K$ resources and yield a revenue of f_j. Over time, the algorithm will decide whether or not to accept customers of type j. Afterwards, a customer type j_t is drawn from an independent distribution. If the algorithm decided to accept customers of type j_t, the relevant resources are consumed and revenue earned. The goal of the decision-maker is to maximize the expected revenue.

This is modeled as an Exo-MDP where $\mathcal{X} = [0, B_1] \times \ldots \times [0, B_K], \Xi = [M]$, and $\mathcal{A} = \{0, 1\}^M$. The system space $\mathcal{X}$ corresponds to the remaining capacity of the K different resources, exogenous space Ξ to the sampled customer type, and $\mathcal{A}$ to the accept / reject decisions for each of the customer types. The reward is then defined via:

$$r(x_t, a, \xi_t) = \begin{cases} f_{\xi_t} & a_{\xi_t} = 1 \text{ and } x_t - A_{\xi_t} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The transition distribution is updated by accounting for consumed resources if a request is accepted:

$$f(x_t, a, \xi_t) = \begin{cases} x_t - A_{\xi_t} & a_{\xi_t} = 1 \text{ and } x_t - A_{\xi_t} \geq 0 \\ x_t & \text{otherwise} \end{cases}$$

C.2.5. Virtual Machine Allocation ($k = T$)

The Cloud has modified the way that users are able access computing resources (Sheng et al., 2021; Jiang et al., 2020; Cortez et al., 2017; Hubbs et al., 2020; Hadary et al., 2020). Cloud service providers allow customers easy access to resources while simultaneously applying efficient management techniques in order to optimize their return. One of the most critical components is the Virtual Machine (VM) allocator, which assigns VM request to the physical hardware (henceforth referred to as PMs). The important issue is how to allocate physical resources to service each VM efficiently by eliminating fragmentation, performance impact and delays, and allocation failures.

These VM allocation models can be thought of as a multi-dimensional variant of bin-packing with an additional component of arrival and departures. The typical VM scheduling scenarios models users requesting resources over time, where each request contains the required CPU and memory uses, and its lifetime. The allocator then decides which available physical machine to allocate the virtual machine to. To limit notational overload, we provide a high level view of the Virtual Machine allocation scenario here, and defer concrete discussion and notation when discussing the planning oracle required for solving.

In this set-up the current system state of the model is measured by the physical resources available for each PM, including the physical cores and memory available. Over time,

- Coming VM requests ask for a certain amount of resources (CPU and memory requirements) and their lifetime. Resource requirements are varied based on the different VM requests.
- Based on the action selected by the algorithm, the VM will be allocated to and be created in a specified PM as long as that PM's remaining resources are enough.
- After a period of execution, the VM completes its tasks. The simulator will then release the resources allocated to this VM and deallocate this VM from the PM.

At a high level, these problems can be modeled as an MDP with exogenous inputs where the exogenous space Ξ contains the space of possible VM requests (along with their lifetime, memory, and CPU requirements). The endogenous space $\mathcal{X}$ measures the current capacity of each physical machine on the server, and action space $\mathcal{A}$ for allocation decisions for the current VM request to a given PM. More details on the concrete experimental set-up will be in Appendix G. We also note that the VM arrival process in practice is highly correlated so this fits under the model where $k = T$ (Hadary et al., 2020).

D. Hindsight Planners

Here we outline the feasibility of Assumption 1 in many operations tasks. Planning problems induced by online knapsack problems as a function of a deterministic input sequence $\boldsymbol{\xi}$ can be solved via their induced linear relaxation in pseudo-polynomial time (as the constraint polytope is a polymatroid). Other problems, such as inventory control with lead times have planning problems where a simple greedy control policy is optimal. More generally, Assumption 1 requires us to solve large-scale combinatorial optimization problems. However, we note that all of these computations are done *offline* and so the computational burden is not required at run-time. Moreover, it is easy to incorporate existing heuristics from the optimization literature for efficient solutions to these problems, including linear programming or fluid relaxations (Conforti et al., 2014). This appeals to our Hindsight Learning algorithms only relying on the objective value of the planner, instead of the actual sequence of actions.

D.1. Inventory Control with Lead Times and Lost Sales

In inventory control, given knowledge of the exact sequence of demands $\boldsymbol{\xi} = (d_1, \ldots, d_T)$, the optimal open loop control policy is trivial to write down. Indeed, setting:

$$a_t = \begin{cases} d_{t+L} & t \leq T - L \\ 0 & \text{otherwise} \end{cases}$$

is clearly the optimal policy. This is as, for any $t \leq T - L$ we ensure the current on-hand inventory is exactly equal to that period's demands. For the last L periods we order nothing in order to minimize the accumulated purchase costs for inventory which will be ordered and cannot be sold.

D.2. Online Stochastic Bin Packing

We give the integer programming representation of the optimal open loop control for Bin Packing as follows. Consider a state x with components $x_1, \ldots, x_B$ as the current number of bins opened with a utilization of 1 up to B and a sequence of items with sizes $\boldsymbol{\xi} = (u_1, \ldots, u_T)$. Given $(x_1, \ldots, x_B)$ we pre-process this list to a vector of length $\sum_i x_i$ where each component corresponds to the current utilization of any bin. For example, if $B = 3$ and $x = (1, 0, 2)$ then we make a list containing $\alpha = (1, 3, 3)$ for two bins with a utilization of three and one bin with a total utilization of one. Since the total number of bins required will be $\sum_i x_i + T$ we use the variables y_b for $b \in [\sum_i x_i + T]$ to denote an indicator of whether or not bin y_b is currently utilized. We also use variables $z_{v,b}$ to denote whether item $v \in [T]$ is assigned to bin b. Similarly, denote α_b as the current utilization of a bin b. The optimization program can then be written as follows:

$$\begin{aligned} \max_{z,y} \; & -\sum_b y_b \\ \text{s.t.} \; & \sum_b z_{v,b} = 1 \text{ for all } v \in [T] \\ & \sum_v z_{v,b} + \alpha_b \leq B y_b \text{ for all } b \\ & y_b = 1 \text{ for any } b \text{ with } \alpha_b > 0 \end{aligned}$$

The objective corresponds to minimizing the number of utilized bins. The first constraint ensures that each item is assigned to a bin. The second constraint enforces capacity constraints for each bin, and the last constraint ensures that bins are marked as used if they have current utilization on them (i.e. $\alpha_b > 0$).

D.3. Online Secretary

In online secretary, given knowledge of the exact sequence of future candidate qualities the open loop control policy is trivial to write down. Indeed, for any $\boldsymbol{\xi}_{\geq t}$ we denote σ as the ranking function over it such that $\xi_\sigma(1) > \xi_\sigma(2) > \ldots > \xi_\sigma(T-t)$, with ties broken arbitrarily. Then given a remaining number of candidates to accept x, $\textsc{Hindsight}(t, \boldsymbol{\xi}_{\geq t}, x)$ will simply be $\sum_{i=1}^{x} \xi_{\sigma(i)}$. This corresponds to taking the best x-candidates from the future trace $\boldsymbol{\xi}_{\geq t}$.

D.4. Airline Revenue Management

The planning oracle for the airline revenue management problem can be formulated as a so-called "knapsack" problem. Indeed, suppose that x_t is the remaining capacity for the K resources. We use variables z_y for the number of customers of type $y \in [M]$ to accept. Then, we solve the following optimization problem:

$$\begin{aligned} \max_{z} \; & \sum_y f_y z_y \\ \text{s.t.} \; & 0 \leq z_y \leq N_y(\boldsymbol{\xi}) \text{ for all } y \in [M] \\ & Az \leq x_t \end{aligned}$$

where $N_y(\boldsymbol{\xi}) = \sum_t \mathbb{1}_{[\xi_t = y]}$ is the number of type y customers in the exogenous dataset $\boldsymbol{\xi}$.

D.5. Virtual Machine Allocation

The planning oracle for the VM allocation problem can be formulated as a large-scale mixed integer linear program. In this section we discuss approaches which utilize this fact in developing an oracle for the hindsight planning problem.

Use $\Xi = V$ to denote the set of VM requests and P as the set of physical machines. We use the following constants which depend on the current inventory of virtual and physical machines contained in the current state s_t, including:

- $\alpha_{t,p}$ for the remaining CPU cores for physical machine p at event t
- $\beta_{t,p}$ for the remaining memory for physical machine p at event t
- $\texttt{CPU-CAP}_p$ and $\texttt{MEM-CAP}_p$ the CPU and memory capacity of physical machine $p \in P$
- $\texttt{LIFETIME}_v$, $\texttt{CORE}_v$, $\texttt{MEM}_v$ as the lifetime, cores, and memory utilization of VM $v \in V$
- $\eta_{v,t}$ an indicator that the VM v is active at time t (i.e. $\eta_{v,t}$ is equal to one for any time starting from the time the VM v arrives until the end of its lifetime)

With this we introduce variables $x_{v,p}$ for each virtual machine v and physical machine p to indicate the assignments. We also use variables $y_{t,p}$ which encode whether physical machine p has a VM assigned to it at time step t.

We start by considering the various constraints in the problem:

- **Assignment Constraint** (Equation (13)): For every v we need $\sum_p x_{v,p} = 1$ indicating that each virtual machine is assigned to a physical machine.
- **CPU Capacity Constraint** (Equation (14)): For every p and t we need $\alpha_{p,t} + \sum_v \texttt{CORE}_v \eta_{v,t} x_{v,p} \leq \texttt{CPU-CAP}_p$ to ensure CPU usage capacity constraints are satisfied.
- **Memory Capacity Constraint** (Equation (15)): For every p and t we need $\beta_{p,t} + \sum_v \texttt{MEM}_v \eta_{v,t} x_{v,p} \leq \texttt{MEM-CAP}_p$ to ensure memory capacity constraints are satisfied.

We also need additional constraints which encode the $y_{t,p}$ variable as follows:

- **PM Historical Utilization** (Equation (16)): $y_{t,p} \geq 1$ for all p and t if $\alpha_{t,p} > 0$
- **PM VM Utilization** (Equation (17)): $x_{v,p}\eta_{v,t} \leq y_{t,p}$ for all v and p
- **PM OR Constraint** (Equation (18)): $\sum_v x_{v,p}\eta_{v,t} + \mathbb{1}_{[\alpha_{t,p}>0]} \geq y_{t,p}$ for all t and p.

These constraints essentially encode that $y_{t,p}$ is an indicator for whether y has a VM assigned to it from $\boldsymbol{\xi}$ (in the second bullet), or has historical allocations on it (for when $\alpha_{t,p} > 0$).

We note that there always exists a feasible solution since we are in an over provisioned regime where we have capacity to service every VM request.

Lastly, the objective function (Equation (12)) for the packing density can be formulated via:

$$-\sum_t \frac{\sum_p y_{t,p} \texttt{CPU-CAP}_p}{\sum_p \alpha_{t,p} + \sum_v \texttt{CORE}_v}$$

The numerator corresponds to the total CPU capacity of all physical machines which are in use. The denominator corresponds to the total utilization (both from the VMs currently in service and the VMs arriving over the time horizon). This then encodes the inverse of the core packing density, as described earlier.

The full integer program is now summarized below:

$$\max_{x,y} -\sum_{t\in[T]} \frac{\sum_{p\in P} y_{t,p} \texttt{CPU-CAP}_p}{\sum_{p\in P} \alpha_{t,p} + \sum_{v\in V} \texttt{CORE}_v} \tag{12}$$

$$\text{s.t.} \sum_{p \in P} x_{v,p} = 1 \quad \forall v \in V \tag{13}$$

$$\alpha_{p,t} + \sum_{v \in V} \texttt{CORE}_v \eta_{v,t} x_{v,p} \leq \texttt{CPU-CAP}_p \quad \forall t \in [T],\, p \in P \tag{14}$$

$$\beta_{p,t} + \sum_{v \in V} \texttt{MEM}_v \eta_{v,t} x_{v,p} \leq \texttt{MEM-CAP}_p \quad \forall t \in [T],\, p \in P \tag{15}$$

$$y_{t,p} \geq \mathbb{1}_{[\alpha_{t,p} > 0]} \quad \forall t \in [T],\, p \in P \tag{16}$$

$$x_{v,p} \eta_{v,t} \leq y_{t,p} \quad \forall v \in V,\, t \in [T],\, p \in P \tag{17}$$

$$\sum_{v \in V} x_{v,p} \eta_{v,t} + \mathbb{1}_{[\alpha_{t,p} > 0]} \geq y_{t,p} \quad \forall t \in [T],\, p \in P \tag{18}$$

E. Existing Approaches to MDPs in Exo-MDPs

Here we briefly discuss existing approaches to MDPs applied in the context of Exo-MDPs to highlight the advantages and disadvantages of our Hindsight Learning approach.

E.1. Predict Then Optimize

Given the historical trace dataset $\mathcal{D} = \{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^N\}$, a popular plug-in approach learns a generative model $\overline{\mathcal{P}_\Xi}(\xi_t \mid \boldsymbol{\xi}_{<t})$ to approximate the true distribution $\mathcal{P}_\Xi(\xi_t \mid \boldsymbol{\xi}_{<t})$, since the exogenous process is the only unknown. Given this model $\overline{\mathcal{P}_\Xi}$, estimates for the $Q_t^\star$ value for the optimal policy can be obtained by solving the Bellman equation with the learned predictor $\overline{\mathcal{P}_\Xi}$ in place of the true distribution $\mathcal{P}_\Xi$. More concretely, we denote $\overline{Q}_t$ as the model-based estimate of $Q_t^\star$, which follows

$$\begin{aligned}
\overline{Q}_t(s,a) &\coloneqq \mathbb{E}_{\xi \mid \boldsymbol{\xi}_{<t}}[r(x,a,\xi) + \overline{V}_{t+1}(f(x,a,\xi) \mid \overline{\mathcal{P}_\Xi}] \\
\overline{V}_t(s) &\coloneqq \max_{a \in \mathcal{A}} \overline{Q}_t(s,a) \\
\overline{\pi}_t(s) &\coloneqq \arg\max_{a \in \mathcal{A}} \overline{Q}_t(s,a).
\end{aligned}$$

While intuitive, this ML forecast approach requires high-fidelity modeling of the exogenous process to guarantee good downstream decision-making, due to the quadratic horizon multiplicative factor in regret we show below. This quadratic factor in the horizon is due to the compounding errors of distribution shift, similar to those shown in the imitation learning literature (Ross et al., 2011).

Theorem 6. *Suppose that* $\sup_{t \in [T], \boldsymbol{\xi}_{<t} \in \Xi^{[t-1]}} \|\overline{\mathcal{P}_\Xi}(\cdot|\boldsymbol{\xi}_{<t}) - \mathcal{P}_\Xi(\cdot|\boldsymbol{\xi}_{<t})\|_{TV} \leq \epsilon$ *where* $\|\cdot\|_{TV}$ *is the total variation distance. Then we have that* $\textsc{Regret}(\overline{\pi}) \leq 2T^2\epsilon$. *In addition, if* $\boldsymbol{\xi} \sim \mathcal{P}_\Xi$ *has each* ξ_t *independent from* $\boldsymbol{\xi}_{<t}$, $\overline{\mathcal{P}_\Xi}$ *is the empirical distribution, then* $\forall \delta \in (0,1)$, *with probability at least* $1-\delta$, $\textsc{Regret}(\overline{\pi}) \leq 2T^2\sqrt{\frac{\log(2|\Xi|/\delta)}{N}}$.

The T^2 dependence here is tight (see Domingues et al. (2021)), in contrast to the $O(T)$ dependence in Theorem 3. Moreover, the ML forecast approach can be impractical when the exogenous process is complex since ϵ in the worst case can scale as $|\Xi|^T$ if the ξ_t are strongly correlated across t. An example of this is VM allocation where researchers observed that the VM lifetime varies substantially across time, the demand has spikes and a diurnal pattern, and that subsequent requests are highly correlated (Hadary et al., 2020).

This discrepancy highlights an advantage of our Hindsight Learning approach. Consider a VM allocation example with two physical machines each large enough to satisfy the entire demand. Under chaotic and unpredictable arrivals, a planner using erroneous forecasts might spread the requests over the two machines. In contrast, the hindsight learning policy will correctly learn that one machine is sufficient and achieve low regret, even if the total variation distance on the underlying distribution over exogenous inputs is large.

E.2. Reinforcement Learning

Recall that our objective is to solve $\arg\max_{\pi \in \Pi} V_1^\pi(s_1)$. This can be written as $\arg\max_{\pi \in \Pi} \mathbb{E}_{\boldsymbol{\xi}}[V_1^\pi(s_1, \boldsymbol{\xi})]$ by Lemma 1. Therefore, an alternative way to find approximately optimal policies for an Exo-MDP is to maximize the empirical return

directly, similar to the empirical risk minimization strategy of supervised learning: $\overline{\pi}^\star = \arg\max_{\pi\in\Pi} \overline{\mathbb{E}}[V_1^\pi(s_1, \boldsymbol{\xi})]$ where $\overline{\mathbb{E}}[V_1^\pi(s_1, \boldsymbol{\xi})] = \frac{1}{N}\sum_n V_1^\pi(s_1, \boldsymbol{\xi}^n)$. First observe that the number of samples required to learn a near optimal policy scales linearly with T in this approach. If an additive control variate $\phi(\boldsymbol{\xi})$ (as in Mao et al. (2019b)) is used, the T term is replaced with $T - \mathbb{E}_{\boldsymbol{\xi}}[\phi(\boldsymbol{\xi})]$.

Theorem 7. *$\forall\delta \in (0,1)$, with probability at least $1-\delta$,* $\textsc{Regret}(\overline{\pi}^\star) \leq T\sqrt{\frac{2\log(2|\Pi|/\delta)}{N}}$.

Theorem 7 highlights that model-free RL methods are a *theoretically viable* approach for Exo-MDPs (especially compared to Predict-Then-Optimize in Theorem 6). Indeed, Theorem 7 shows that RL methods have asymptotic consistency guarantees *if* they converge to the optimal policy in the empirical MDP. This convergence is an idealized computation assumption that hides optimization issues when studying statistical guarantees, and is incomparable to Assumption 1 for the hindsight planner (for which we showed several examples in Appendix D.

Moreover, Table 1 suggests that HL and RL are trading bias and variance differently. When variance is the dominating factor, we expect an algorithm's performance to improve with additional data. When bias is the dominating factor, however, we expect no marginal benefit from additional data. In Table 1 we see that as T (and accordingly, N, the number of data points) increases, the Tabular RL algorithm performance improves. However, hindsight learning has a stable non-zero regret even as we increase T.

F. Proofs of Main Results

Lemma 8 (Lemma 1 of §3)**.** *For every $t \in [T]$, $(s,a) \in \mathcal{S}\times\mathcal{A}$, and $\pi \in \Pi$, we have the following:*

$$Q_t^\pi(s,a) = \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[Q_t^\pi(s,a,\boldsymbol{\xi}_{\geq t})]$$
$$V_t^\pi(s) = \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\pi(s,\boldsymbol{\xi}_{\geq t})].$$

In particular $V_1^\pi(s_1) = V_1^\pi = \mathbb{E}_{\boldsymbol{\xi}}[V_1^\pi(s_1,\boldsymbol{\xi})]$.

Proof. First note that if $Q_t^\pi(s,a)$ is as defined then we have that:

$$\begin{aligned}
V_t^\pi(s) &= \sum_a \pi(a \mid s) Q_t^\pi(s,a) \text{ (by Bellman equations)} \\
&= \sum_a \pi(a \mid s)\mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[Q_t^\pi(s,a,\boldsymbol{\xi}_{\geq t})] \\
&= \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}\left[\sum_a \pi(a \mid s) Q_t^\pi(s,a,\boldsymbol{\xi}_{\geq t})\right] \text{ (by } \pi \text{ being non-anticipatory)} \\
&= \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\pi(s,\xi_{\geq t})].
\end{aligned}$$

Now we focus on showing the result for $Q_t^\pi(s,a)$ by backwards induction on t. The base case when $t = T$ is trivial as $Q_T^\pi(s,a) = \mathbb{E}_\xi[r(s,a,\xi)] = \mathbb{E}_{\boldsymbol{\xi}_{\geq T}}[Q_T^\pi(s,a,\boldsymbol{\xi}_{\geq T})]$.

Step Case: $(t+1 \to t)$ For the step-case a simple derivation shows that

$$\begin{aligned}
Q_t^\pi(s,a) &= \mathbb{E}_\xi\left[r(s,a,\xi) + V_{t+1}^\pi(f(s,a,\xi))\right] \\
&= \mathbb{E}_\xi\left[r(s,a,\xi) + \mathbb{E}_{\boldsymbol{\xi}_{\geq t+1}}\left[V_{t+1}^\pi(f(s,a,\xi),\boldsymbol{\xi}_{\geq t+1})\right]\right] \\
&= \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}\left[r(s,a,\xi_t) + V_{t+1}^\pi(f(s,a,\xi_t),\xi_{\geq t+1})\right] \\
&= \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[Q_t^\pi(s,a,\xi_{\geq t})].
\end{aligned}$$

□

Theorem 9 (Theorem 2 of §6)**.** $\textsc{Regret}(\pi^\dagger) \leq \sum_{t=1}^T \mathbb{E}_{S_t\sim \Pr_t^{\pi^\dagger}}[\Delta_t^\dagger(S_t)]$ *where $\Pr_t^{\pi^\dagger}$ denotes the state distribution of $\pi^\dagger$ at step t induced by the exogenous randomness. In particular, if $\Delta_t^\dagger(s) \leq \Delta$ for some constant Δ then we have:* $\textsc{Regret}(\pi^\dagger) \leq \Delta\sum_{t=1}^T \mathbb{E}_{S_t\sim \Pr_t^{\pi^\dagger}}[\Pr(\pi_t^\dagger(S_t) \neq \pi^\star(S_t))]$.

Proof. First we note that via the performance difference lemma we have that for any two non-anticipatory policies π and $\tilde{\pi}$ that

$$V_1^{\pi}(s_1) - V_1^{\tilde{\pi}}(s_1) = \sum_t \mathbb{E}_{s_t \sim \Pr_t^{\pi}} \left[\sum_a \pi(a \mid s_t)(Q_h^{\tilde{\pi}}(s,a) - V_h^{\tilde{\pi}}(s)) \right] \text{ and so}$$
$$V_1^{\tilde{\pi}}(s_1) - V_1^{\pi}(s_1) = \sum_t \mathbb{E}_{s_t \sim \Pr_t^{\pi}} \left[\sum_a \pi(a \mid s_t)(V_h^{\tilde{\pi}}(s) - Q_h^{\tilde{\pi}}(s,a)) \right]$$

Moreover, for any state s we also have $Q_t^{\star}(s, \pi^{\star}(s)) - Q_t^{\star}(s, \pi^{\dagger}(s)) \leq \Delta_t^{\dagger}(s)$ since:

$$\begin{aligned} &Q_t^{\star}(s, \pi^{\star}(s)) - Q_t^{\star}(s, \pi^{\dagger}(s)) - \Delta_t^{\dagger}(s) \\ &= Q_t^{\star}(s, \pi^{\star}(s)) - Q_t^{\star}(s, \pi^{\dagger}(s)) - Q_t^{\dagger}(s, \pi^{\dagger}(s)) + Q_t^{\star}(s, \pi^{\dagger}(s)) - Q_t^{\star}(s, \pi^{\star}(s)) + Q_t^{\dagger}(s, \pi^{\star}(s)) \\ &= Q_t^{\dagger}(s, \pi^{\star}(s)) - Q_t^{\dagger}(s, \pi^{\dagger}(s)) \leq 0 \end{aligned}$$

as $\pi^{\dagger}$ is greedy with respect to $Q^{\dagger}$.

Finally, recall the definition of the regret of $\textsc{Regret}(\pi^{\dagger})$ via $V_1^{\star}(s_1) - V_1^{\pi^{\dagger}}(s_1)$. However, using the previous performance difference lemma with $\tilde{\pi} = \pi^{\star}$ and $\pi = \pi^{\dagger}$ we have that

$$\begin{aligned} V_1^{\star}(s_1) - V_1^{\pi^{\dagger}}(s_1) &= \sum_t \mathbb{E}_{S_t \sim \Pr_t^{\pi^{\dagger}}} \left[Q_t^{\star}(S_t, \pi^{\star}(S_t)) - Q_t^{\star}(S_t, \pi^{\dagger}(S_t)) \right] \\ &\leq \sum_t \mathbb{E}_{S_t \sim \Pr_t^{\pi^{\dagger}}} \left[\Delta_t^{\dagger}(S_t) \right]. \end{aligned}$$

The second statement follows immediately from the absolute bound on $\Delta_t^{\dagger}(s)$. □

Theorem 10 (Theorem 3 of §6)**.** *Let $\overline{\pi}^{\dagger}$ denote the hindsight planning surrogate policy for the empirical MDP w.r.t. $\mathcal{D}$. Assume $\overline{\pi}^{\dagger} \in \Pi$ and Algorithm 1 achieves no-regret in the optimization problem. Let π be the best policy generated by Algorithm 1. Then, for any $\delta \in (0,1)$, with probability $1-\delta$, it holds*

$$\textsc{Regret}(\pi) \leq T \sqrt{\frac{2\log(2|\Pi|/\delta)}{N}} + \sum_{t=1}^{T} \mathbb{E}_{s_t \sim \overline{Pr}_t^{\overline{\pi}^{\dagger}}} [\bar{\Delta}_t^{\dagger}(s_t)] + o(1)$$

where $\bar{\Delta}_t^{\dagger}$ is the SAA approximation of (11) and $\overline{Pr}_t^{\overline{\pi}^{\dagger}}$ is the state probability of $\overline{\pi}^{\dagger}$ in the empirical MDP.

Proof. The proof follows the standard proof technique of online IL (cf. (Yan et al., 2021)).

$$\begin{aligned} \textsc{Regret}(\pi) &= \mathbb{E}[V_1^{*}(s_1, \boldsymbol{\xi})] - \mathbb{E}[V_1^{\pi}(s_1, \boldsymbol{\xi})] \\ &\leq \left(|\overline{\mathbb{E}}[V_1^{\pi}(s_1, \boldsymbol{\xi})] - \mathbb{E}[V_1^{\pi}(s_1, \boldsymbol{\xi})]| + |\overline{\mathbb{E}}\left[V_1^{\pi^*}(s_1, \boldsymbol{\xi})\right] - \mathbb{E}\left[V_1^{\pi^*}(s_1, \boldsymbol{\xi})\right]| \right) \\ &\quad + \left(\overline{\mathbb{E}}\left[V_1^{\pi^*}(s_1, \boldsymbol{\xi})\right] - \overline{\mathbb{E}}\left[V_1^{\overline{\pi}^{\dagger}}(s_1, \boldsymbol{\xi})\right] + \overline{\mathbb{E}}\left[V_1^{\overline{\pi}^{\dagger}}(s_1, \boldsymbol{\xi})\right] - \overline{\mathbb{E}}[V_1^{\pi}(s_1, \boldsymbol{\xi})] \right) \\ &\leq 2T \sqrt{\frac{2\log(2|\Pi|/\delta)}{N}} + \overline{\mathbb{E}}\left[V_1^{\pi^*}(s_1, \boldsymbol{\xi})\right] - \overline{\mathbb{E}}\left[V_1^{\overline{\pi}^{\dagger}}(s_1, \boldsymbol{\xi})\right] + o(1) \\ &\leq 2T \sqrt{\frac{2\log(2|\Pi|/\delta)}{N}} + \sum_{t=1}^{T} \mathbb{E}_{s_t \sim \bar{\Pr}_t^{\overline{\pi}^{\dagger}}} [\bar{\Delta}_t^{\dagger}(s_t)] + o(1) \end{aligned}$$

We use the results of Theorem 15 to bound the first terms in the second line; we invoke the no-regret optimization assumption and the realizability assumption $\overline{\pi}^{\dagger} \in \Pi$ for the last term of the second line. Finally, we apply Theorem 2 in the empirical MDP w.r.t. $\mathcal{D}$ and recognize that the middle term is the empirical regret to derive the last step. □

Lemma 11 (Theorem 4 of §6)**.** *There exists a set of Exo-MDPs such that $\textsc{Regret}(\pi^{\dagger}) \geq \Omega(T)$.*

Proof. We first construct a three-step MDP such that $\textsc{Regret}(\pi^\dagger) \geq \Omega(1)$. The main result then follows by replicating the MDP across T periods to construct a $3T$ step MDP with $\textsc{Regret}(\pi^\dagger) \geq \Omega(T)$.

We consider a modification of the prototypical Pandora's Box problem (Weitzman, 1979). The endogenous state space $\mathcal{X} = \{0, 1\}$ where state 0 corresponds to "not yet accepted an item" and 1 corresponds to "accepted an item". The action space $\mathcal{A} = \{0, 1\}$ where $a = 0$ corresponds to "reject next item" and $a = 1$ corresponds to "accept next item". We consider a modification of the typical Pandora box model where at time step t, the next item arrivals ξ_t value is unobserved before deciding whether or not to accept.

The trace distribution has $\xi_1 \sim U[0, 1]$ and $\xi_2, \xi_3 \sim U[0, .9]$. Important to note is that $\mathbb{E}[\xi_1] = .5, \mathbb{E}[\xi_2] = 0.45, \mathbb{E}[\xi_3] = 0.45$, and a straightforward calculation shows that $\mathbb{E}[\max(\xi_2, \xi_3)] = 0.6$.

The rewards and dynamics are:

$$\begin{aligned} r(0, 0, \xi) &= 0 & f(0, 0, \xi) &= 0 \\ r(0, 1, \xi) &= \xi & f(0, 1, \xi) &= 1 \\ r(1, a, \xi) &= 0 & f(1, a, \xi) &= 1 \text{ for } a \in \{0, 1\}. \end{aligned}$$

Lastly, the starting state $s_1 = 0$. This properly encodes the exogenous dynamics and rewards. At step t in state $x = 0$ (i.e. not yet accepted an item) taking action 0 (do not accept) yields no return and transitions to the next state. However, accepting the next item returns reward ξ and transitions to state $x = 1$.

A straightforward calculation following the Bellman equations shows the following for $Q_t^\star$ and $V_t^\star$:

$$\begin{aligned} Q_3^\star(0, 0) &= 0 & Q_2^\star(0, 0) &= \mathbb{E}[\xi_3] & Q_1^\star(0, 0) &= \max(\mathbb{E}[\xi_2], \mathbb{E}[\xi_3]). \\ Q_3^\star(0, 1) &= \mathbb{E}[\xi_3] & Q_2^\star(0, 1) &= \mathbb{E}[\xi_2] & Q_1^\star(0, 1) &= \mathbb{E}[\xi_1] \\ Q_3^\star(1, \cdot) &= 0 & Q_2^\star(1, \cdot) &= 0 & Q_1^\star(1, \cdot) &= 0 \\ V_3^\star(0) &= \mathbb{E}[\xi_3] & V_2^\star(0) &= \max(\mathbb{E}[\xi_3], \mathbb{E}[\xi_2]) & V_1^\star(0) &= \max(\mathbb{E}[\xi_1], \mathbb{E}[\xi_2], \mathbb{E}[\xi_3]) \\ V_3^\star(1) &= 0 & V_2^\star(1) &= 0 & V_1^\star(1) &= 0. \end{aligned}$$

Using the choice of the distributions for ξ_1, ξ_2, ξ_3 we have that $\pi_1^\star(0) = 1$, as in, we will accept the first item since on average it has larger expected return. This results in $V_1^{\pi^\star}(s_1) = \mathbb{E}[\xi_1] = 0.5$.

We can similarly compute $Q_t^\dagger$ and $V_t^\dagger$ as follows:

$$\begin{aligned} Q_3^\dagger(0, 0) &= 0 & Q_2^\dagger(0, 0) &= \mathbb{E}[\xi_3] & Q_1^\dagger(0, 0) &= \mathbb{E}[\max(\xi_2, \xi_3)]. \\ Q_3^\dagger(0, 1) &= \mathbb{E}[\xi_3] & Q_2^\dagger(0, 1) &= \mathbb{E}[\xi_2] & Q_1^\dagger(0, 1) &= \mathbb{E}[\xi_1] \\ Q_3^\dagger(1, \cdot) &= 0 & Q_2^\dagger(1, \cdot) &= 0 & Q_1^\dagger(1, \cdot) &= 0. \end{aligned}$$

In this scenario, we see first hand the bias introduced when considered $Q^\dagger$. In particular, $Q_1^\dagger(0, 0) = \mathbb{E}[\max(\xi_2, \xi_3)] \geq Q_1^\star(0, 0) = \max(\mathbb{E}[\xi_2, \xi_3])$. Using the choice of distributions for ξ_1, ξ_2, ξ_3 we see that the hindsight planning policy $\pi^\dagger$ is fooled and has $\pi_1^\dagger(0) = 0$, so the policy rejects the first item thinking it will get the maximum value of the next two items. As a result we see that $V_1^{\pi^\dagger}(0) = 0.45$.

Hence, we have that $\textsc{Regret}\pi^\dagger = 0.5 - 0.45 = 0.05 = \Omega(1)$ as needed. □

Lemma 12 (Statement in §6)**.** *Suppose that for every t and state s that $\max_{\pi \in \Pi} \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\pi(s, \boldsymbol{\xi}_{\geq t})] = \mathbb{E}_{\boldsymbol{\xi}_{>t}}[\textsc{Hindsight}(t, s, \boldsymbol{\xi}_{\geq t})]$. Then we have that $Q_t^\star(s, a) = Q_t^\dagger(s, a)$ for every t, s, and a.*

Proof. First notice that $\max_{\pi \in \Pi} \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\pi(s, \boldsymbol{\xi}_{\geq t})] = \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\star(s, \boldsymbol{\xi}_{\geq t})]$ by definition. Thus using the Bellman equations and definition of $Q_t^\dagger(s, a)$ we trivially have that:

$$\begin{aligned} Q_t^\star(s, a) &= \mathbb{E}_{\xi_t}[r(s, a, \xi_t) + V_{t+1}^\star(f(s, a, \xi_t))] \\ &= \mathbb{E}_{\xi_t}[r(s, a, \xi_t) + \mathbb{E}_{\boldsymbol{\xi}_{>t}}[V_{t+1}^\star(f(s, a, \xi_t), \boldsymbol{\xi}_{>t})]] \\ &= \mathbb{E}_{\xi_t}[r(s, a, \xi_t) + \mathbb{E}_{\boldsymbol{\xi}_{>t}}[\textsc{Hindsight}(t + 1, f(s, a, \xi_t), \boldsymbol{\xi}_{>t})]] \end{aligned}$$

$$= Q_t^\dagger(s, a).$$

□

Lemma 13 (Statement in §6). *Define* $\widetilde{GAG} = \mathbb{E}\Big[\max_\pi \sum_{t=1}^T \Delta_t^\dagger(S_t) \mid S_t \sim \Pr_t^\pi\Big]$. *Then there exists an Exo-MDP such that* $\widetilde{GAG} = \Omega(T)$ *and yet the upper bound in Theorem 2 is zero.*

The "MakeDecision" function in Mercier & Van Hentenryck (2007) implements the empirical Bayes Selector policy using the offline dataset (see Equation (6)). There are several key differences between our regret analysis and theirs. First, Mercier & Van Hentenryck (2007) define regret with respect to the hindsight optimal policy $V^\dagger(s_1)$, whereas our regret is with respect to the best non-anticipatory policy $V^\star(s_1)$. Their main result (Theorem 1) shows that the hindsight optimal regret is bounded by the "Global Anticipatory Gap" $GAG = \mathbb{E}\Big[\max_\pi \sum_{t=1}^T \Omega_t(S_t, A_t) \mid (S_t, A_t) \sim \Pr_t^\pi\Big]$. However, there are situations where the GAG can be large, and one could incorrectly conclude that hindsight optimization should not be applied when trying to learn the true optimal non-anticipative policy $\pi^\star$. In the Sailing example of §4, the GAG will be positive since knowing the direction of the wind one can ex-post identify the optimal route compared to any non-anticipatory algorithm. However, $Q^\star(\text{route}) = Q^\dagger(\text{route})$ and so there is no hindsight bias. Hence, our results which adjust for the difference in benchmark to $\pi^\star$ is more appropriate.

Moreover, their regret bound measures a worst-case overestimation bias on the states visited by any decision policy. In contrast, Theorem 2 requires that the hindsight bias be small only on states visited by $\pi^\dagger$. Even if we set aside the difference between our benchmarks for regret (adjusting the definition by $V^\star(s_1) - V^\dagger(s_1)$ resulting in $\widetilde{GAG}$ defined in the statement of Lemma 13), our analysis is much tighter which we illustrate with an example below.

Proof. We construct an Exo-MDP $\mathcal{M}$ as follows. Consider a starting state s_0 with two actions A and B that deterministically transition to two different "sub-MDPs" (which we denote as $\mathcal{M}_A$ and $\mathcal{M}_B$ respectively).

The first action, A, transitions to sub-MDP $\mathcal{M}_A$ which contains an absorbing state s_A (i.e. transitions are deterministic to the same state s_A regardless of the action) with large rewards. Note that since this sub-MDP is deterministic it has no hindsight bias (so $\Delta_t^\dagger(s_A) = 0$).

The second action B, transitions to an MDP $\mathcal{M}_B$ which witnesses Theorem 4, and hence has $\Omega(T)$ hindsight bias. We adjust the rewards along $\mathcal{M}_B$ such that the value of the optimal policy in this sub-MDP is much smaller than the deterministic reward accrued in $\mathcal{M}_A$.

In this example, $\pi^\dagger$ will always select action A in the initial state and collect higher rewards (since the optimal policy knowing the exogenous inputs in MDP $\mathcal{M}_B$ will still collect smaller rewards than the deterministic value accrued in $\mathcal{M}_A$). Hence, our regret bound in Theorem 2 will be zero (since $\Delta^\dagger(s_A) = 0$ and $\pi^\dagger$ will never visit s_B). However, $\widetilde{GAG}$ will conservatively account for the sub-optimal B decision which transitions to a state with a large anticipatory gap ($\Omega(T)$), thereby concluding a large regret bound. □

Theorem 14 (Theorem 6 of Appendix E). *Suppose that* $\sup_{t\in[T], \boldsymbol{\xi}_{<t} \in \Xi^{[t-1]}} \|\overline{\mathcal{P}_\Xi}(\cdot|\boldsymbol{\xi}_{<t}) - \mathcal{P}_\Xi(\cdot|\boldsymbol{\xi}_{<t})\|_{TV} \leq \epsilon$ *where* $\|\cdot\|_{TV}$ *is the total variation distance. Then we have that* $\textsc{Regret}(\overline{\pi}) \leq 2T^2\epsilon$. *In addition, if* $\boldsymbol{\xi} \sim \mathcal{P}_\Xi$ *has each* ξ_t *independent from* $\boldsymbol{\xi}_{<t}$, $\overline{\mathcal{P}_\Xi}$ *is the empirical distribution, then* $\forall \delta \in (0, 1)$, *with probability at least* $1 - \delta$, $\textsc{Regret}(\overline{\pi}) \leq 2T^2\sqrt{\frac{\log(2|\Xi|/\delta)}{N}}$.

Proof. First note that $\overline{Q}_t$ and $\overline{V}_t$ refer to the Q and V values for the optimal policy in a modified MDP $\overline{M}$ where the true exogenous input distribution $\mathcal{P}_\Xi(\cdot \mid \boldsymbol{\xi}_{<t})$ is replaced by its estimate $\overline{\mathcal{P}_\Xi}(\cdot \mid \boldsymbol{\xi}_{<t})$. As such, denote by $\overline{V}_t^\pi$ as the value function for the policy π in the MDP $\overline{M}$. Note here that $\overline{V}_t^{\overline{\pi}} = \overline{V}_t$ by construction. With this we have that:

$$\begin{aligned}
\textsc{Regret}(\overline{\pi}) &= V_1^\star(s_1) - V_1^{\overline{\pi}}(s_1) \\
&= V_1^\star(s_1) - \overline{V}_1^{\pi^\star}(s_1) + \overline{V}_1^{\pi^\star}(s_1) - \overline{V}_1(s_1) + \overline{V}_1(s_1) - V_1^{\overline{\pi}}(s_1) \\
&\leq 2 \sup_\pi |V_1^\pi(s_1) - \overline{V}_1^\pi(s_1)|.
\end{aligned}$$

However, using the finite horizon simulation lemma (see Lemma 1 in Abbeel & Ng (2005)) we have that this is bounded from above by $2T^2\|P(\cdot \mid s, a) - \overline{P})(\cdot \mid s, a)\|_{TV}$ where $\|P(\cdot \mid s, a) - \overline{P})(\cdot \mid s, a)\|_{TV}$ is the total variation distance in the

induced state-transition distributions between M and $\overline{M}$. However, by definition we have that:

$$\begin{aligned}\|P(\cdot \mid s,a) - \overline{P}(\cdot \mid s,a)\|_{TV} &= \frac{1}{2}\int_{\mathcal{S}} |P(s' \mid s,a) - \overline{P}(s' \mid s,a)| ds \\ &= \frac{1}{2}\int_{\mathcal{S}} |\int_{\Xi} \mathbb{1}_{[s'=f(s,a,\xi)]} d\mathcal{P}_{\Xi}(\xi \mid \boldsymbol{\xi}_{\geq t}) - \mathbb{1}_{[s'=f(s,a,\xi)]} d\overline{\mathcal{P}_{\Xi}}(\xi \mid \boldsymbol{\xi}_{\geq t})| ds \\ &= \frac{1}{2}\int_{\mathcal{S}}\int_{\Xi} \mathbb{1}_{[s'=f(s,a,\xi)]} |d\mathcal{P}_{\Xi}(\xi \mid \boldsymbol{\xi}_{\geq t}) - d\overline{\mathcal{P}_{\Xi}}(\xi \mid \boldsymbol{\xi}_{\geq t})| \\ &\leq \|\mathcal{P}_{\Xi}(\cdot \mid \boldsymbol{\xi}_{\geq t}) - \overline{\mathcal{P}_{\Xi}}(\cdot \mid \boldsymbol{\xi}_{\geq t})\|_{TV} \leq \epsilon.\end{aligned}$$

Thus we get that $\textsc{Regret}(\overline{\pi}) \leq 2T^2\epsilon$ as required.

Now suppose that $\boldsymbol{\xi} \sim \mathcal{P}_{\Xi}$ has each ξ_t independent from $\boldsymbol{\xi}_{<t}$ and let $\overline{\mathcal{P}_{\Xi}}$ be the empirical distribution, i.e. $\overline{\mathcal{P}_{\Xi}}(\xi) = \frac{1}{N}\sum_{i\in[N]} \mathbb{1}_{[\boldsymbol{\xi}_t^i = \xi]}$. A straightforward application of Hoeffding's inequality shows that the event:

$$\mathcal{E} = \left\{ \forall t, \boldsymbol{\xi}_{<t} : |\overline{\mathcal{P}_{\Xi}}(\Xi) - \mathcal{P}_{\Xi}(\xi)| \leq \sqrt{\frac{\log(2|\Xi|/\delta)}{2N}} \right\}$$

occurs with probability at least $1-\delta$. Under $\mathcal{E}$ we then have that:

$$\sup_{t\in[T], \boldsymbol{\xi}_{<t}\in\Xi^{[t-1]}} \|\overline{\mathcal{P}_{\Xi}}(\cdot|\boldsymbol{\xi}_{<t}) - \mathcal{P}_{\Xi}(\cdot|\boldsymbol{\xi}_{<t})\|_{TV} \leq \sup_{t\in[T], \boldsymbol{\xi}_{<t}\in\Xi^{[t-1]}} \|\overline{\mathcal{P}_{\Xi}}(\cdot|\boldsymbol{\xi}_{<t}) - \mathcal{P}_{\Xi}(\cdot|\boldsymbol{\xi}_{<t})\|_{1} \leq \sqrt{\frac{\log(2|\Xi|/\delta)}{2N}}.$$

Taking this in the previous result shows the claim. □

Theorem 15 (Theorem 7 of Appendix E)**.** *Given any $\delta \in (0,1)$ then with probability at least $1-\delta$ we have that if $\overline{\pi}^\star = \arg\max_\pi \overline{\mathbb{E}}[V^\pi(\xi)]$ that*

$$\textsc{Regret}(\overline{\pi}^\star) \leq \sqrt{\frac{2T^2\log(2|\Pi|/\delta)}{N}}.$$

Proof. A quick calculation using Hoeffding's inequality and a union bound shows that the event

$$\mathcal{E} = \left\{ \forall \pi \in \Pi : |V_1^\pi(s_1) - \overline{\mathbb{E}}[V_1^\pi(s_1)]| \leq \sqrt{\frac{T^2\log(2|\Pi|/\delta)}{2N^2}} \right\}$$

occurs with probability at least $1-\delta$. Under $\mathcal{E}$ we then have that:

$$\begin{aligned}\textsc{Regret}(\overline{\pi}^\star) &= V_1^{\pi^\star}(s_1) - V_1^{\overline{\pi}^\star}(s_1) \\ &= V_1^{\pi^\star}(s_1) - \overline{\mathbb{E}}\left[V_1^{\pi^\star}(s_1,\boldsymbol{\xi})\right] + \overline{\mathbb{E}}\left[V_1^{\pi^\star}(s_1,\boldsymbol{\xi})\right] - \overline{\mathbb{E}}\left[V_1^{\overline{\pi}^\star}(s_1,\boldsymbol{\xi})\right] + \overline{\mathbb{E}}\left[V_1^{\overline{\pi}^\star}(s_1,\boldsymbol{\xi})\right] - V_1^{\overline{\pi}^\star}(s_1) \\ &\leq 2\sqrt{\frac{V_{max}^2\log(2|\Pi|/\delta)}{2N^2}}.\end{aligned}$$

□

Theorem 16 (Lemma 18 of Appendix G.1)**.** *In stochastic online bin packing with i.i.d. arrivals we have that $\sup_{t,s}\Delta_t^\dagger(s) \leq O(1)$, independent of the time horizon and any problem primitives. As a result, $\textsc{Regret}(\pi^\dagger) \leq O(1)$.*

We show the result by starting with the lemma, highlighting that the value functions for the planning policy and the optimal non-anticipatory policy are "Lipschitz" with respect to the capacity of the current bins. Recall that the state space representation $s \in \mathcal{S}$ corresponds to $s = (x, \xi_{t-1})$ where $x \in \mathbb{R}^B$ is the current number of bins at that size, and the last component to the current arrival. We write this explicitly as containing $s \in \mathbb{R}^B$ for the bin utilization and $\xi_{t-1} \in \mathbb{R}$ for the current arrival.

Lemma 17. *For any $t \in [T]$, current bin capacity $x \in \mathbb{R}^{|B|}$, current arrival ξ_{t-1}, $\boldsymbol{\xi}_{\geq t} \in \Xi^{T-t}$, and $\Delta \in \mathbb{R}^B \geq 0$ we have that:*

- $V_t^\dagger(x, \xi_{t-1}, \boldsymbol{\xi}_{\geq t}) \geq V_t^\dagger(x - \Delta, \xi_{t-1}, \boldsymbol{\xi}_{\geq t}) \geq V_t^\dagger(x, \xi_{t-1}, \boldsymbol{\xi}_{\geq t}) - \|\Delta\|_1$
- $V_t^\star(x, \xi_{t-1}) \geq V_t^\star(x - \Delta, \xi_{t-1}) \geq V_t^\star(x, \xi_{t-1}) - \|\Delta\|_1$

As a result for any x and x' in $\mathbb{R}^B$ and current arrival ξ_{t-1} we have that:

- $V_t^\dagger(x, \xi_{t-1}, \boldsymbol{\xi}_{\geq t}) - V_t^\dagger(x', \xi_{t-1}, \boldsymbol{\xi}_{\geq t}) \leq \|(x - x')^+\|_1$
- $V_t^\star(x, \xi_{t-1}) - V_t^\star(x', \xi_{t-1}) \leq \|(x - x')^+\|_1$

Proof. First consider the top statement in terms of the optimal planning policy starting from a fixed state $s = (x, \xi_{t-1})$ and sequence of future exogenous variables $\boldsymbol{\xi}_{\geq t}$.

We have that $V_t^\dagger(x, \xi_{t-1}, \boldsymbol{\xi}_{\geq t}) \geq V_t^\dagger(x - \Delta, \xi_{t-1}, \boldsymbol{\xi}_{\geq t})$ since the sequence of actions generated by the planning oracle starting from state $(x - \Delta, \xi_{t-1})$ is feasible for the same problem starting from (x, ξ_{t-1}). Hence, as $V_t^\dagger(x, \xi_{t-1}, \boldsymbol{\xi}_{\geq t})$ denotes the optimal such policy, the inequality follows.

For the other direction consider the sequence of actions starting from (x, ξ_{t-1}). Using at most $\|\Delta\|_1$ bins the policy is feasible for the same problem starting at $(x - \Delta, \xi_{t-1})$. Indeed, suppose the sequence of actions starting from the problem at (x, ξ_{t-1}) attempts to use a bin which is not available in the problem starting from $(x - \Delta, \xi_{t-1})$. Then by opening a new bin instead and shifting all future references of the old bin to the newly created bin, the sequence of actions is feasible. As there are at most $\|\Delta\|_1$ bins different in the (x, ξ_{t-1}) problem versus the $(x - \Delta, \xi_{t-1})$ problem, the bound follows.

Now consider the second statement in terms of the optimal non-anticipatory policy starting from a fixed state (x, ξ_{t-1}). First note that $V_t^\star(x, \xi_{t-1}) = \mathbb{E}_{\boldsymbol{\xi}_{\geq t}}[V_t^\star(x, \xi_{t-1}, \boldsymbol{\xi}_{\geq t})]$ and similarly for $V_t^\star(x - \Delta, \xi_{t-1})$. We have that $V_t^\star(x, \xi_{t-1}) \geq V_t^\star(x - \Delta, \xi_{t-1})$ as the optimal policy starting from $(x - \Delta, \xi_{t-1})$ is feasible on all sample paths generated by $\boldsymbol{\xi}_{\geq t}$ to the same problem starting at (x, ξ_{t-1}). Hence by optimality of $\pi^\star$ the inequality must follow.

For the other direction, on any sample path consider the sequence of actions generated by the optimal policy starting from (x, ξ_{t-1}). By a similar argument, again using at most $\|\Delta\|_1$ extra bins the policy is feasible for the problem starting at $(x - \Delta, \xi_{t-1})$. Hence by optimality, the inequality follows.

The second result follows via straightforward algebraic manipulations. Indeed, the previous statement can be thought of as showing that for $x \in \mathbb{R}^B$ and $\Delta \in \mathbb{R}^B_+$ that $f(x) \geq f(x - \Delta) \geq f(x) - \|\Delta\|_1$. However,

$$f(x) - f(x') = f(x) - f(x' + (x - x')^+) + f(x' + (x - x')^+) - f(x') \leq f(x' + (x - x')^+) - f(x') \leq \|(x - x')^+\|_1$$

where the first inequality uses that $x' + (x - x')^+ \geq x$ and the second the previous result. □

We are now ready to show the bound that $\Delta_t^\dagger(s) \leq O(1)$.

Proof. For a fixed time t and state s consider $\Delta_t^\dagger(s) = Q_t^\dagger(s, \pi^\dagger(s)) - Q_t^\star(s, \pi^\dagger(s)) + Q_t^\star(s, \pi^\star(s)) - Q_t^\dagger(s, \pi^\star(s))$

However consider $Q_t^\dagger(s, \pi^\dagger(s)) - Q_t^\dagger(s, \pi^\star(s))$ (with the other term dealt with similarly). On any sample path, the difference in these terms is bounded by the immediate reward plus the difference of the value at the next states. By problem definition, the difference in immediate rewards is bounded by one. However, consider the difference in value functions at the next state. Their state representation has a value of $\|(x - x')^+\|_1$ of at most 2 (for the two bins that were potentially modified). Hence, this difference is bounded by 3 in total. A similar argument for $Q^\star$ completes the proof. □

G. Simulation Details

In this section we provide full details on the simulations conducted, including a formal description of virtual machine allocation scenarios along with its fidelity to the real-world cloud scenarios, training implementations, hyperparameter tuning results, and a description of the heuristic algorithms compared.

G.1. Online Bin-Packing

In a **Stochastic Online Bin Packing (OBP)** problem the agent has an infinite supply of bins of size B. Each round, items $u_t \in \{0, \dots B\}$ arrive sampled iid from an unknown distribution. The agent either *packs* the item into an opened feasible bin or *opens* a new bin, with the goal to minimize the expected number of opened bins after T rounds.

Appendix C describes how OBP are Exo-MDPs with u_t as the exogenous inputs. Lemma 3.1 from Freund & Banerjee (2019) shows that in stochastic OBP, $\Pr(\pi^\dagger_t(S_t) \neq \pi^\star_t(S_t)) \leq O(\frac{1}{t^2})$. Intuitively, as $t \to T$ there is little contribution from $\textsc{Hindsight}(t, \boldsymbol{\xi}_{>t})$ to the $Q^\dagger$ of Equation 6 and so $\pi^\dagger$ is more likely to coincide with $\pi^\star$. Using a novel absolute bound on Δ and Theorem 2, we have:

Lemma 18. *In stochastic online bin packing with i.i.d. arrivals,* $\sup_{t,s} \Delta^\dagger_t(s) \leq O(1)$*, independent of any problem primitives. Hence,* $\textsc{Regret}(\pi^\dagger) \leq O(1)$.

To numerically validate this claim, we use `ORSuite` (Archer et al., 2022) as an OBP simulator with $B = 5$ and vary T from 5 to 100. For these small problem sizes, we can compute $\pi^\star$ by solving Bellman equations exactly. The exogenous process $\mathcal{P}_\Xi$ is uniform: $u_t \sim \text{Unif}[B]$ and we generate $|\mathcal{D}| = 1000$ traces, and benchmark the resulting learned policy against $\pi^\star$. The hindsight planner is represented with the integer program in Appendix D.2 (solved efficiently by linear relaxation). Any learned policy maps a $B + 2$-dim state (a vector denoting number of bins open with utilization from 1 to B, the current item and the remaining rounds) to a decision $A \in \{0, \dots, B - u_t\}$ to select a feasible bin (0 opens a new bin).

Table 5: Hindsight bias in OBP decreases as T increases. Thus, $\pi^\dagger$ becomes a better surrogate for $\pi^\star$.

T	MaxBias	$\%\ \{\exists s : \pi^*_t(s) \neq \pi^\dagger_t(s)\}$
5	1.240	6.8%
10	0.646	3.4%
100	0.066	0.3%

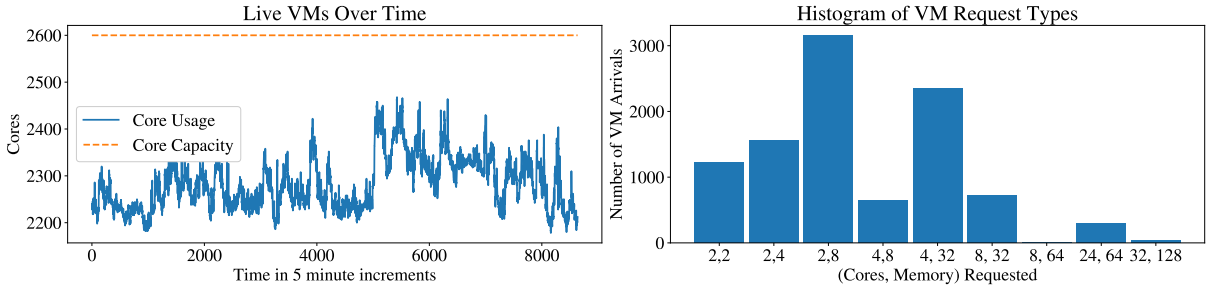
For each OBP problem with $T \in \{5, 10, 100\}$ we report in Table 5 the maximum hindsight bias $\text{MaxBias} = \mathbb{E}\big[\max_{s,t} \Delta^\dagger_t(s)\big]$, where the expectation averages over 1000 sampled problem instances. We also report the percentage of problem instances where at least a single state witnesses $\pi^*_t(s) \neq \pi^\dagger_t(s)$. We see that as T increases, $\pi^\dagger$ becomes a good surrogate for $\pi^\star$ which bodes well for HL.

G.2. Multi-Secretary

Multi-secretary is the generalization of the classic secretary problem (Buchbinder et al., 2009), where T candidates arrive sequentially but only B can be selected. An arriving candidate at time t has ability $r_t \in (0, 1]$ drawn i.i.d. from a finite set of K levels of expertise. At each round, if the decision-maker has remaining budget (i.e., has chosen less than B candidates thus far), they can *accept* a candidate and collect the reward r_t, or *reject* the candidate. The goal is to maximize the expected cumulative reward. Appendix C.2.3 shows how the multi-secretary problem can be formulated as an Exo-MDP.

We use $T = \{5, 10, 100\}$, $B = \frac{3}{5}T$, $K = 4$ for our experiments. With four expertise levels the corresponding abilities for the expertise levels was chosen to be $\{1/4, 1/2, 3/4, 1\}$. The arrival process for the ability types is non-stationary and sinusoidal with a type-dependent shift and frequency. Denoting p^t_j as the arrival probability of a type j customer at timestep t, the distribution is as follows. First, p^1_j is chosen to be uniformly at random from $[0, 2\pi]$. Next, the frequency for each j is chosen to be uniformly from $[0, \pi/4]$. The final arrival probabilities p^t_j are then chosen to be sinusoidal with that shift and frequency value. These values are then normalized appropriately to be a valid distribution.

In Table 2 we report the performance by evaluating each policy using dynamic programming with the true arrivals distribution. The Greedy heuristic accepts the first B candidates regardless of their quality. ML methods uses a single trace sampled from the non-stationary candidate arrival process, and use a policy that maps a 3-dim state (the rounds and budget remaining, and the current candidate ability) to an *accept* probability. For the hindsight planner, we use Equation 2 from Banerjee et al. (2020) which implements a linear program with $2N$ variables.



(a) Figure A (b) Figure B

Figure 4: Sample of a thirty-day roll-out from the Azure Public Dataset. In Figure 4(b) we show a histogram of the various VM types and their corresponding cores and memory resources requested. In Figure 4(a) we plot the used cores over time on the observed trace, along with the capacity of the 80-node cluster simulated using MARO.

G.3. Airline Revenue Management

Airline Revenue Management (Littlewood, 1972) is a special case of the multi-dimensional Online Bin Packing (OBP) problem (recall that OBP exhibits vanishing hindsight bias via Lemma 18). The agent has capacity B_k for K different resources. At each round, the decision-maker observes a request $A_t \in \mathbb{R}_+^K$ (the consumed capacity in each resource dimension), alongside a revenue f_t. The algorithm can either *accept* the request (obtaining revenue f_t and updating remaining capacity according to A_t), or *reject* it (note that partial acceptance is not allowed). The goal of the decision-maker is to maximize the expected revenue.

We use `ORSuite` (Archer et al., 2022) as an ARM simulator with fixed capacity, iid. request types and job distribution. We use $T = \{5, 10, 100\}, K = 3$, and 2 request types. The starting capacity for the three resources set to be $[8, 4, 4]$. The iid arrival distribution is $(1/3, 1/3, 1/3)$ (where the last category corresponds to no arrival). Job one arrivals have resource requests $[2, 3, 2]$ with revenue 1, and job two arrivals have resource requests $[3, 0, 1]$ with revenue 2. This setting satisfies a dual-degeneracy condition of the hindsight planner from Vera & Banerjee (2021) which shows large regret for existing heuristics on these problems.

The optimal policy is computed through dynamic programming. Both RL (Tabular Q-learning) and HL (Hindsight MAC) were trained on the same dataset, which contained 100 traces. In Table 2 we report the performance of the policies through Monte Carlo simulations averaged over 500 iterations.

G.4. Virtual Machine Allocation

Cloud computing has revolutionized the way that computing resources are consumed. These providers give end-users easy access to state-of-the-art resources. One of the most crucial components to cloud computing providers is the Virtual Machine (VM) allocator, which assigns specific VM requests to physical hardware. Improper placement of VM requests to physical machines (henceforth referred to as PMs) can cause performance impact, service delays, and create allocation failures. The VMs serve as the primary units of resource allocation in these models. We focus on designing allocation policies at the *cluster* level, which are a homogeneous set of physical machines with the same memory and CPU cores capacity.

The cluster-specific allocator is tasked with the following:

- Coming VM requests ask for a certain amount of resources (CPU and memory requirements) along with their lifetime. Resource requirements are varied based on the different VM requests.
- Based on the action selected by the allocator policy, the VM will be allocated to and be created in a specified PM as long as that PM's remaining resources are enough.
- After a period of execution, the VM completes its tasks. The simulator will then release the resources allocated to this VM and de-allocate this VM from the PM.

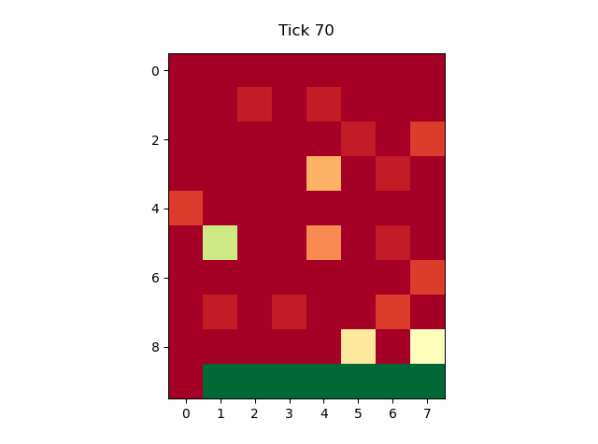


Figure 5: Packing density for the Best Fit policy at one time-step. Each square corresponds to a specific PM and the colour corresponds to what portion of that PMs capacity is currently utilized. Red corresponds to fully used, green is completely empty. The packing density ignores the empty (or green) PMs on the bottom and counts the cumulative utilization ratio for the remaining PMs.

G.5. Stylized Environment

We use `MARO`, an open source package for multi-agent reinforcement learning in operations research tasks as a simulator for the VM allocator (Jiang et al., 2020). In this scenario the VM requests are uniformly sampled from the 2019 snapshot of the Azure Public Dataset (Cortez et al., 2017). The cluster is a fictitious one consisting of 80 PMs that we found were similarly over-provisioned as in real-world clusters. See Figure 4(a) to highlight the demand workload against the cluster capacity for our experiment setup.

By default, `MARO` provides reward metrics that can be used when specifying the objective of the algorithm. The metrics provided include income, energy consumption, profit, number of successful and failed allocations, latency, and total number of overloaded physical machines. However, typical cloud computing systems run in an *over-provisioned* regime where the capacity of the physical machines is larger than the demand in order to ensure quality of service to its customers. As a result, any reasonable algorithm has no failed allocations. Hence, any reasonable algorithm also has identical values for income, energy consumption, profit, etc.

For the reward function we instead consider $r(s, a, \xi) = -100 * \texttt{Failed-Allocation} - 1/\texttt{Packing Density}$. The first component, one hundred times the number of failed allocations, helps to penalize the algorithms in training to ensure valid assignments for all of the VM requests. The second component corresponds to the `Packing-Density`, computed via:

$$\texttt{Packing-Density} = \frac{\sum_{v \in VM} \texttt{Cores Usage}_v}{\sum_{p \in PM} \mathbb{1}_{[p \text{ is utilized}]} \texttt{Capacity}_p}$$

The numerator here is the total cumulative cores used for all of the VMs currently assigned on the system. The denominator is the total capacity of all physical machines which are currently utilized (i.e. have a VM assigned and currently running). The reason for picking this reward (and the inverse of it) is that:

- It allows for an easily expressible linear programming formulation (see Appendix D.5).
- For any two policies which allocate all virtual machines, packing density serves as a criteria to differentiate the policies. An algorithm which has large packing density equivalently uses the physical machines efficiently so that unused PMs can be re-purposed, reassigned, or potentially turned off.
- It serves as a proxy to ensure that the virtual machines are packed in as minimal number of physical machines possible. This allows the allocator to be robust to hardware failures, where entire physical machines are potentially rendered unusable.

Algorithm 2 Training Procedure in MARO

1: **Input:** number of roll-outs, number of actors, number of training iterations.
2: **for** each roll-out **do**
3: For each actor, sample a random duration uniformly at random, and execute the **Best Fit** heuristic on a historical dataset of that length starting from an empty cluster
4: For each actor, sample a one-hour trace of VM requests $\boldsymbol{\xi}^i$
5: **for** each actor **do**
6: Collect dataset of $(s_t, a_t, \xi_t, r_t, s_{t+1})$ pairs under the current policy π_θ
7: **end for**
8: Add collected dataset to current experience buffer
9: **for** each training iteration **do**
10: Sub-sample batch from current collected dataset
11: Update policy by gradient descent along the sampled batch
12: **end for**
13: **end for**

See Figure 5 for an illustration.

G.5.1. Simulator Fidelity

Our training procedure requires a faithful simulator of Virtual Machine allocation focused at a cluster level to validate our experimental results. We found that the MARO simulator captures all first-order effects of cloud computing environments specifically at the cluster level. However, there are several effects not included in the simulations:

- When a virtual machine arrives to the system, they request a *maximum* amount of CPU and memory capacity that they can use. However, over time, any given VM request might only use a fraction of their requested resources. Current cloud computing systems use an *over-subscription* model where the requested memory and CPU cores for the VMs assigned to a PM can surpass its capacity. However, when the total realized demand surpasses the PMs capacity, all of the VMs assigned to that system are failed and migrated to a different PM. In contrast, MARO assumes that each VM uses exactly its requested cores and memory over time, hence eliminating the need to model over-subscription on the cluster level.
- Typical systems involve live migration where virtual machines can be moved between physical machines without disrupting the VM request. This is used in order to eliminate stranding which occurs when a physical machine has only a few long-running virtual machines allocated to it. However, such an operation is costly and requires a large amount of system overhead.
- Our neural networks explicitly use the VM's lifetime as a feature in the state. However, in true cloud computing systems the lifetime is unknown. Only when a user decides to cancel a VM does the system have access to that information. As such, it is typically observed in the trace dataset but cannot be used in policy network design. We forgo this when modelling the policy as unlike the dynamics of VM request types, lifetimes for a VM are typically easy to model and these forecasts can be used as a replacement in the network representation (Hadary et al., 2020).

We believe that MARO serves as a high fidelity simulator of the VM allocation problem at the cluster level while providing open source implementation for additional research experimentation. However, we complement these results in §7.3.2 with a real-world model of cloud computing platforms.

G.5.2. Training Implementation Details

Algorithm 2 presents the pseudocode for our training procedure using MARO. We repeat the following process for five hundred roll-outs. First, we created realistic starting state distributions by executing the *BestFit* heuristic for a random duration (greater than one day). Line 4 samples one-hour traces of VM requests from the 2019 historical Microsoft Azure dataset. Then, in line five and six, with fifteen actors in parallel we evaluate the current policy π_θ on the sampled VM request trace, adding the dataset of experience to the experience buffer. Lines 9-11 samples batches of size 256 where we update the policy and algorithmic parameters θ by gradient descent on the loss function evaluated on the sampled batch. This process repeats for five thousand gradient steps.

We implemented the training framework using PyTorch (Paszke et al., 2019) along with `MARO` (Jiang et al., 2020), and all experiments were conducted using the Microsoft Azure ML training platform. For hyperparameters and neural network architectures for the Sim2Real RL and Hindsight Learning algorithms, see Appendix G.5.4. All experiments were run on the same compute hardware and took similar runtimes to finish. Runtime was dominated by the `MARO` simulator executing the roll-outs under the curent policy versus the HINDSIGHT calls or the ML model updates. As such, each algorithm was essentially given the same computational budget.

To evaluate the trained policies we subdivided the Azure Public Dataset into a temporally contiguous and non-overlapping training (first 15 days) and test (last 15 days) portions. For evaluation we sampled fifty different one-day traces of VM requests from the held-out portion. For each of the different traces, we executed the policy in parallel to a greedy **Best Fit** algorithm. Each deep learning algorithm was evaluated over five different random seed initializations and we tuned hyperparameters using grid search. All metrics are reported with respect to cumulative differences against the baseline **Best Fit** policy, alongside statistical significance tests. In Table 2 we evaluate the number of machines required to pack the jobs. Negative numbers correspond to fewer required PMs on average. Asterisks correspond to statistical significance computed with Welch's t-test with $p = 0.05$. In Table 7 we provide the performance metrics on the underlying rewards as well.

G.5.3. HEURISTIC ALGORITHMS

- **Random**: Picks a physical machine uniformly at random from the list of physical machines which have capacity to service the current VM request.
- **Round Robin**: Allocates to physical machines in a round-robin strategy by selecting a physical machine from the list of physical machines which have capacity to service the current VM request that was least recently used.
- **Best Fit**: This algorithm picks a physical machine based on a variety of metric types.
  - **Remaining Cores**: Picks a valid physical machine with minimum remaining cores
  - **Remaining Memory**: Picks a valid physical machine with minimum remaining memory
  - **Energy Consumption**: Picks the valid physical machine with maximum energy consumption
  - **Remaining Cores and Energy Consumption**: Picks a valid physical machine with minimum remaining cores, breaking ties via energy consumption

  Similar heuristics to this are currently used in most large-scale cloud computing systems (Hadary et al., 2020).
- **Bin Packing**: Selects a valid physical machine which minimizes the resulting variance on the number of virtual machines on each physical machine.

G.5.4. SIM2REAL RL ALGORITHMS

We also compared our Hindsight Learning approaches to existing Sim2Real RL algorithms in the literature with custom implementation built on top of the `MARO` package. These include:

- **Deep Q Network (DQN)**: Double Q-Learning algorithm from Van Hasselt et al. (2016).
- **Actor Critic (AC)**: Actor Critic algorithm implementation from Konda & Tsitsiklis (2000).
- **Mean Actor Critic (MAC)**: A modification of the actor critic algorithm where the actor loss is calculated along all actions instead of just the selected actions (Allen et al., 2017).
- **Policy Gradient (VPG)**: A modification of the vanilla policy gradient with a exogenous input dependent baseline from Mao et al. (2019b). Instead of training a baseline explicitly, we use $Q_t^{\text{Best Fit}}(s, a, \boldsymbol{\xi}_{\geq t})$.

G.5.5. STATE FEATURES, NETWORK ARCHITECTURE, AND HYPERPARAMETERS

The state space of the VM allocation scenario is combinatorial as we need to include the CPU and memory utilization of each physical machine across time to account for the lifetimes of the VMs currently assigned to the PM. To rectify this, for each of the deep RL algorithms we use action-dependent features when representing the state space. In particular, once a VM request arrives, we consider the set of physical machines that are available to service this particular virtual machine. Each (PM, VM) pair has associated state features, including:

- The VM's CPU cores requirement, memory requirement, and lifetime
- The PM's CPU cores capacity, memory capacity, and type
- The historical CPU cores allocated, utilization, energy consumption, and memory allocated over last three VM requests

The last component is serving as a proxy for the historical utilization of the PM across all time to account for all VMs currently assigned to the PM. The final action dependent features corresponds to the concatenation of these state features for each valid PM to service the current request.

We note that to use action-dependent features some of the algorithms required slight tweaking to their implementations. In particular, when considering algorithms using a policy network representation (i.e. policy gradient, actor critic, or mean actor critic) when executing the policy we take $\pi_\theta = \text{Softmax}(\pi(s, a_1), \ldots, \pi(s, a_N))$ where $a_1, \ldots, a_N$ is the set of physical machines that can service the current request and (s, a_i) is the corresponding state-features for the physical machine a_i.

For the actor and critic network representations in all algorithms we use a four layer neural network with $(32, 16, 8)$ hidden dimensions, an output dimension of one (due to the action-dependent features), and LeakyReLU activation functions. For each of the algorithms we use the *RMSprop* optimization algorithm. We implemented the training framework using PyTorch (Paszke et al., 2019) and `MARO`. All experiments were run on the same compute hardware and took similar runtimes to finish. Runtime was dominated by the `MARO` simulator executing the roll-outs, and the ML model updates and HINDSIGHT oracle calls were quicker.

Lastly we provide a list of the hyperparameters used and which algorithm they apply to when tuning algorithm performance.

Table 6: List of hyperparameters tuned over for the Sim2Real RL and Hindsight Learning algorithms.

Hyperparameter	Algorithm	Values
Discount Factor	DQN, AC, MAC, PG	0.9, 0.95, 0.99, 0.999
Learning Rate	All Algorithms	0.05, 0.005, 0.0005, 0.00005, 0.000005
Entropy Regularization	PG, AC, MAC	0, 0.1, 1, 10
Actor Loss Coefficient	AC, MAC	0.1, 1, 10, 100
Target Update Smoothing Parameter	DQN	0.0001, 0.001, 0.01, 0.1

For concrete parameters evaluated and experiment results, see the attached code-base.

G.5.6. TRAINING PERFORMANCE

In Figure 6 we include a plot of the loss curves for the various algorithms.

G.5.7. TESTING PERFORMANCE

In Table 7 we plot the performance of the algorithms on the true reward function. Here we include the true reward considered:

$$r(s, a, \xi) = -100 * \texttt{Failed-Allocation} - 1/\texttt{Packing Density}(s)$$

as well as simply $\texttt{Packing Density}(s)$. All measures are reported with a 95% confidence interval, and computed as differences against the **Best Fit** allocation policy.

G.6. Real-World VM Allocation

In this simulation we consider the more realistic setting of VM arrivals in continuous time and where the allocation agent has no information about VM lifetimes. We avoid giving a concrete description of the reward function trained, cluster sizes, etc. to preserve the confidentiality of the cloud operator. However, we briefly describe the training implementation details, heuristic algorithms, as well as the hindsight heuristic used.

G.6.1. HEURISTIC ALGORITHMS

Table 3 summarizes the results of performance for three different algorithms:

Table 7: Performance of heuristics, Sim2Real RL, and Hindsight Learning algorithms on VM allocation. Here we include the true reward metric the algorithms were trained on (negative inverse of the packing density) and the packing density improvements on average.

Algorithm	Performance $r = -1/\texttt{Packing Density}$	Packing Density
Performance Upper Bound	0.66 ± 0.29	$.09\% \pm 0.03\%$
Best Fit	0.0	0.0
Bin Packing	-64.44 ± 2.49	$-5.34\% \pm 0.2\%$
Round Robin	-56.36 ± 2.65	$-4.67\% \pm 0.22\%$
Random	-48.94 ± 2.33	$-4.08\% \pm 0.19\%$
DQN	-1.00 ± 0.41	$-0.05\% \pm 0.04\%$
MAC	-0.38 ± 0.033	$-0.03\% \pm 0.00\%$
AC	-2.94 ± 0.61	$-0.21\% \pm 0.06\%$
Policy Gradient	-1.03 ± 0.39	$-0.06\% \pm 0.04\%$
HINDSIGHT MAC	$\mathbf{0.18 \pm 0.093}$	$\mathbf{0.05\% \pm 0.00\%}$
HINDSIGHT Q-DISTILLATION	0.08 ± 0.32	$0.04\% \pm 0.029\%$

Algorithm 3 Hindsight Heuristic.

1: **Input:** A cluster state s, sequence of VM requests $\boldsymbol{\xi}_{t:T}$.
2: Sort requests in descending order of their lifetimes
3: **for** Each request ξ **do**
4: Allocate to the feasible PM where ξ is the only live VM on it for the least amount of time
5: **end for**

BestFit Performance is shown relative to a **BestFit** strategy used in production. This strategy follows a proprietary implementation that prioritizes between CPU and memory depending on their scarcity.

Hindsight Learning and Sim2Real RL We adapt HINDSIGHT MAC (HL) and compare it with MAC (Allen et al., 2017) (RL), where both used the same network architecture, which embeds VM-specific and PM-specific features using a 6-layer GNN. The resulting architecture is rich enough to represent the **BestFit** heuristic, but can also express more flexible policies.

G.6.2. TRAINING IMPLEMENTATION DETAILS

We trained each algorithm over 3 random seeds and evaluated 5 rollouts to capture the variation in the cluster state at the start of the evaluation trace. Unlike the other experiments, we cannot account for the randomness in exogenous samples because we only have one evaluation trace for each cluster. Error metrics are computed with a paired t-test of value $p = 0.05$.

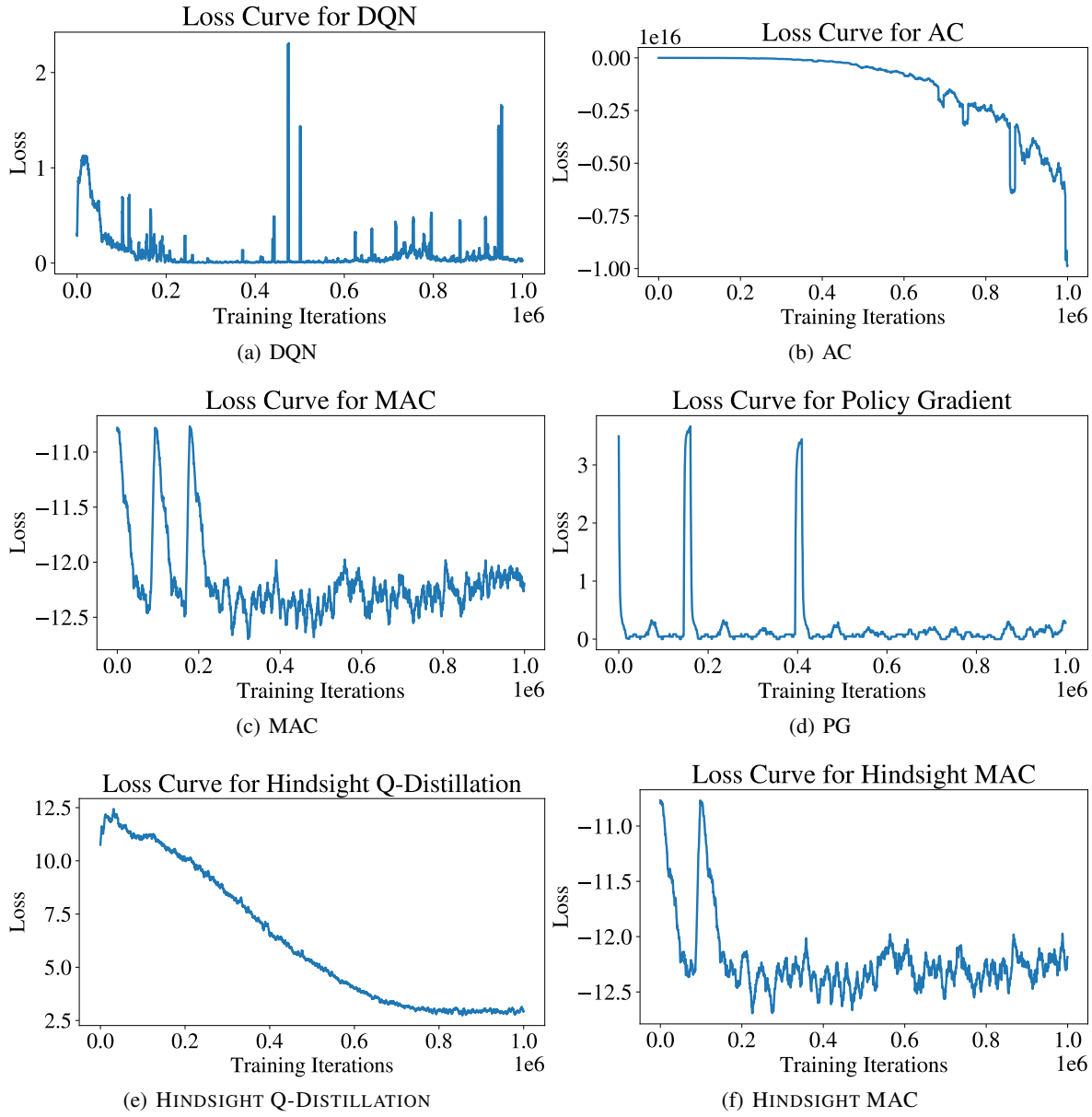
G.6.3. HINDSIGHT HEURISTIC

Due to the large scale of the real-world scenarios, even the linear relaxation of the integer program was not tractable. Consequently, we resort here to using a carefully designed *hindsight heuristic* (Algorithm 3) to derive $\text{HINDSIGHT}(t, \boldsymbol{\xi}, s)$. The heuristic is based on prioritizing VMs according to both their size and duration. See Algorithm 3 for pseudocode.

In Table 8 we separately tested the accuracy of our heuristic for the hindsight planner (both by comparing to the optimal in small instances, as well as by comparing to a lower bound given in Buchbinder et al. (2021)), and concluded that the heuristic obtains a value that is within few percentages of the optimum. We found that the dual gap of Algorithm 3 was typically within 4% of the optimum.

Table 8: How close to optimal is the Upper Bound (Oracle)? We measure the average UsedPMs of the Oracle's solution and compare with the lower bound which assumes that VMs can be fractionally split across PMs.

Cluster	Upper Bound (Oracle)	Lower Bound	Gap (%)
A	467.42	499.625	6.89%
B	538.35	578.214	7.41%
C	383.19	391.329	2.12%
D	27.47	32.9769	20.07%
E	448.86	475.968	6.04%
F	577.97	625.083	8.15%
G	2252.65	2287.21	1.53%
H	2295.19	2332.31	1.62%
I	341.90	361.654	5.78%
J	1212.91	1239.9	2.23%
K	565.34	570.532	0.92%
L	8.23	8.85826	7.64%
M	8.77	9.3152	6.25%
N	305.10	310.096	1.64%
O	43.27	45.3596	4.82%
P	2528.72	2588.38	2.36%
Q	1457.37	1481.13	1.63%
R	123.04	124.866	1.49%
S	2452.25	2491.68	1.61%
T	68.68	70.0956	2.07%
U	533.61	539.872	1.17%
V	1243.70	1260.14	1.32%
W	1678.88	1702.09	1.38%
X	158.71	171.058	7.78%
ALL			4.33%



(a) DQN (b) AC

(c) MAC (d) PG

(e) HINDSIGHT Q-DISTILLATION (f) HINDSIGHT MAC

Figure 6: Moving average of the loss curves for the Sim2Real RL and Hindsight Learning algorithms over the one million gradient steps computed in each of the experiments. Note that some of the volatility occurs when an algorithm observes a datapoint with a failed allocation as there is a large penalty.