Optimal Admission Control for Multiclass Queues with Time-Varying Arrival Rates via State Abstraction

Marc Rigter,[1, 2] **Danial Dervovic,**[2] **Parisa Hassanzadeh,**[2]
Jason Long,[2] **Parisa Zehtabi,** [2] **Daniele Magazzeni** [2]
[1]Oxford Robotics Institute, University of Oxford
[2]J. P. Morgan AI Research
mrigter@robots.ox.ac.uk, {danial.dervovic, parisa.hassanzadeh, jason.x.long, parisa.zehtabi, daniele.magazzeni}@jpmorgan.com

Abstract

We consider a novel queuing problem where the decision-maker must choose to accept or reject randomly arriving tasks into a no buffer queue which are processed by N identical servers. Each task has a *price*, which is a positive real number, and a *class*. Each class of task has a different price distribution, service rate, and arrives according to an inhomogenous Poisson process. The objective is to decide which tasks to accept so that the total price of tasks processed is maximised over a finite horizon. We formulate the problem using a discrete time Markov Decision Process (MDP) with a hybrid state space. We show that the optimal value function has a specific structure, which enables us to solve the hybrid MDP exactly. Moreover, we rigorously prove that as the gap between successive decision epochs grows smaller, the discrete time solution approaches the optimal solution to the original continuous time problem. To improve the scalability of our approach to a greater number of servers and task classes, we present an approximation based on state abstraction. We validate our approach on synthetic data, as well as a real financial fraud data set, which is the motivating application for this work.

Introduction

In many service systems, the rate at which tasks arrive may greatly exceed the capacity of servers to process the tasks. We are motivated by the problem of financial fraud detection and monitoring (Dal Pozzolo et al. 2014), where the rate at which suspicious transactions are detected may considerably exceed the rate at which human operators (servers) can intervene, for example, by holding payments and calling customers to verify their identities.

We associate each task with a *price*, and a *class*. The price of a task represents the value associated with processing that task. In the financial fraud domain, the value of intervening to verify a suspicious transaction may be equal to that transaction's monetary value. Tasks of each class may have different service time requirements and price distributions. For example, validating a high-value bank transfer (task class A) may require a more thorough process, and therefore a slower service rate, than a low-value credit card payment (task class B). In the financial fraud domain, the rate of task arrivals varies substantially with time: the volume of transactions is far greater during business hours than at night. Therefore, we allow the arrival rate of each task class to vary over time.

For such systems, we are interested in *admission control*, the problem of deciding which tasks should be processed, and which tasks should be ignored. We seek to optimise the expected total price of tasks processed over a finite horizon by a fixed number of identical servers.

The optimal control of multiclass queues has been considered by previous works (Ata 2006; Ata and Tongarlak 2013; Bertsimas, Paschalidis, and Tsitsiklis 1994; Cao and Xie 2016; Harrison and Zeevi 2004; Klimov 1974). However, all of these works consider an infinite horizon with constant task arrival rates, which is a considerably easier problem as the optimal solution is stationary (i.e. does not depend on time). Yoon and Lewis (2004) consider optimal admission control in a time-varying problem where there is only one task class, and the task prices can only take on values from a discrete set. The authors present an approximation algorithm, but do not provide any theoretical analysis.

In contrast, we consider multiple task classes with time-varying arrival rates and allow task prices to be continuous. We rigorously prove that our discrete-time algorithm approaches the optimal solution as the time step is reduced. To improve the scalability of our approach to a greater number of task classes, we present an approximation based on state abstraction to reduce the size of the state space.

Our main contributions are: 1) formulating a novel multiclass queuing problem with continuous task prices and time-varying arrival rates, 2) rigorously proving that our discrete-time algorithm approaches the optimal continuous time solution as the time step is reduced, and 3) an approach based on state abstraction to improve scalability. We validate our approach on synthetic domains, as well as a real financial fraud data set. Our results demonstrate that our approach is computationally efficient and significantly improves the average total price of tasks processed compared to baselines.

Related Work

The stochastic sequential assignment problem (SSAP) with random arrival times was proposed by Albright (1974), building on earlier work from Derman, Lieberman, and Ross (1972). In this problem, tasks arrive according to an inhomogenous Poisson process and must be accepted or rejected upon arrival. Furthermore, each task is associated with a continuous reward value drawn from an arbitrary distribution. This problem is also referred to as the dynamic and stochastic knapsack problem (Kleywegt and Papastavrou 1998). The SSAP was recently revisited by Dervovic et al. (2021) to address the problem when the arrival and reward distributions must be learnt from historical data. In contrast to our work, the SSAP problem assumes that the number of tasks to be accepted is known a priori, whereas we assume that tasks have stochastic processing times.

Semi-MDPs were introduced by (Jewell 1963) and (Howard 1963) and have been applied extensively to the optimal control of queues (Rue and Rosenshine 1985; Stidham 1985; Sennott 1989). However, these works assume that the arrival rate of tasks is *stationary*, i.e. does not vary with time, and optimise the expected reward over an infinite horizon. Under these assumptions, the optimal policy is also stationary (Sennott 1989). The queue control problem with time-varying arrival rates that we address is an instance of a non-stationary Semi-MDP (Ghosh and Saha 2013). Finding the exact optimal solution for non-stationary Semi-MDPs is generally not possible as the solution depends on time, and the optimality equations involve integrals over time which are intractable for most problems (McMahon 2008). Mamer (1986) proposes an approximation which limits the number of possible transitions, while while Duckworth, Lacerda, and Hawes (2021) present an approximation algorithm which utilises sample-based planning.

Multiclass queues have multiple task classes which may have different arrival rates and service rates. The optimisation of multiclass queues has been addressed in numerous works (Ata 2006; Ata and Tongarlak 2013; Bertsimas, Paschalidis, and Tsitsiklis 1994; Cao and Xie 2016; Harrison and Zeevi 2004; Klimov 1974). However, none of these existing works consider time-varying arrival rates, which is a core focus of this work.

The most related work to ours is that of Yoon and Lewis (2004). Yoon and Lewis (2004) consider tasks arriving according to a periodic inhomogenous (i.e. time-varying) Poisson process. All tasks have the same service rate, but may take on price values from a discrete set. The authors present a piecewise stationary approximation of the optimal admission policy. The authors do not provide performance guarantees for this approach. In contrast to Yoon and Lewis (2004), our work a) admits continuous task prices values, b) considers multiple task classes with different service rates and time-varying arrival rates, and c) we rigorously prove that our discrete-time algorithm approaches the optimal continuous-time solution as the time step is reduced.

Problem Formulation

Problem 1 (Continuous-time formulation)**.** Consider a multi-server queue with N_{serv} identical servers. A *task* is defined by a pair $\tau = (k, p)$, where $k \in K = \{1, 2, \ldots, |K|\}$ is the *task class*, and $p \in [0, \infty)$ is the *task price*. Tasks of each class arrive according to an inhomogenous Poisson process, $\Lambda_k(t)$. The arrival rate function for each class is assumed to be finite and Lipschitz continuous with constant C_k. The price distribution for each task is independent of the arrival time. The probability density function for the price of task class k is $f_k(p)$, with corresponding cumulative density function, $F_k(p)$. When a new task arrives, the decision maker observes the task class and price, but does not observe the required processing time. If there is at least one free server who is not currently processing a task, the decision-maker can decide to accept or reject the new task. If the new task is accepted, it immediately starts being processed by one of the servers who was previously free. If there are no servers who are free, the decision-maker cannot accept new tasks (i.e. there is no buffer or "waiting area"). For each task class, the time to process a task is exponentially distributed with rate μ_k. The objective is to find a decision-making policy to maximise the expected total price of tasks which are completed during a finite horizon of length t_H.

Discrete-Time Solution

In Problem 1, tasks arrive in continuous time and the rate of arrivals varies continuously with time. To solve Problem 1 exactly, it can be formalised as a Semi-MDP with time-varying dynamics (Ghosh and Saha 2013). However, solving Semi-MDPs with time-varying dynamics exactly is intractable (McMahon 2008). Therefore, we take the approach of discretising time and approximate Problem 1 as a discrete-time hybrid factored MDP (HMDP) with finite horizon. In Proposition 4 we prove that as the resolution of the discretisation of time approaches zero, the solution to our HMDP formulation approaches the solution to Problem 1.

Discrete-Time Hybrid Factored MDPs

A discrete-time hybrid factored MDP (HMDP) (Kveton, Hauskrecht, and Guestrin 2006) with a finite horizon is a tuple, $\mathcal{M} = (\mathbf{X}, A, R, P, D)$. $D = \{t_0, t_1, \ldots, t_H\}$ is a finite sequence of decision epochs, or time steps. For simplicity, we assume that the time step size, $\Delta t = t_{i+1} - t_i$, is a constant. $\mathbf{X}$ is a state space represented by a set of state variables, $\{X_1, X_2, \ldots, X_n\}$. A state is defined by a vector $\mathbf{x}$ of assignments to each state variable, which splits into discrete and continuous components denoted by $\mathbf{x} = (\mathbf{x}_D, \mathbf{x}_C)$. For any state vector, $\mathbf{x}$, we write $\mathbf{x}[j]$ to refer to the value of state variable j. A is a finite set of actions. $R : \mathbf{X} \times A \times D \rightarrow \mathbb{R}_{\geq 0}$ is a non-negative reward function. $P : \mathbf{X} \times A \times D \times \mathbf{X} \rightarrow \mathbb{R}_{\geq 0}$ is a time-varying transition function which describes the transition dynamics conditioned upon the previous state, action, and time step. A deterministic Markovian policy is a mapping from the state and time step to an action: $\pi : \mathbf{X} \times D \rightarrow A$. The objective is to find a policy which optimises the expected total reward over the finite horizon, $\mathbb{E}[\sum_{t=t_0}^{t_H} R(\mathbf{x}_t, \pi(\mathbf{x}_t))]$. The optimal value function (Bellman 1966), V_{π^*}, satisfies

$$V_{\pi^*}(\mathbf{x}, t_i) = \max_{a \in A} \Big[R(\mathbf{x}, a, t_i) + \sum_{\mathbf{x}'_D} \int_{\mathbf{x}'_C} P(\mathbf{x}' \mid \mathbf{x}, a, t_i) \cdot V_{\pi^*}(\mathbf{x}', t_{i+1}) \, \mathrm{d}\mathbf{x}_C \Big], \quad (1)$$

where we denote the optimal policy corresponding to Eq. 1 by π^*. The optimal Q-values are

$$Q_{\pi^*}(\mathbf{x}, a, t_i) = R(\mathbf{x}, a, t_i) + \sum_{\mathbf{x}'_D} \int_{\mathbf{x}'_C} P(\mathbf{x}' \mid \mathbf{x}, a, t_i) \cdot V_{\pi^*}(\mathbf{x}', t_{i+1}) \, \mathrm{d}\mathbf{x}_C. \quad (2)$$

HMDP Formulation

To approximate the continuous-time formulation in Problem 1 by an HMDP, we make the following assumption.

Assumption 1. *At each decision epoch, at most one task can be accepted.*

Assumption 1 is necessary to ensure that the action space is finite. If a task, τ, arrives between t_i and t_{i+1}, at decision epoch t_{i+1} the decision maker can choose to accept or reject τ. In our formulation, we assume that if more than one task arrives between decision epochs, one of these tasks is selected uniformly at random to be the task that the decision maker can choose to accept or reject. The rest of the arrivals are automatically rejected.

We now introduce our HMDP formulation of the problem which we refer to as the Stochastic Task Admission HMDP (STA-HMDP). The STA-HMDP has the following set of state variables: $\{n_1, \ldots, n_{|K|}, k^+, p\}$. n_k is the number of servers who are currently processing tasks of class k. Because there are N_{serv} servers, $\sum_{k \in K} n_k \leq N_{serv}$. If task $\tau = (k_{\texttt{arr}}, p_{\texttt{arr}})$ arrives between t_i and t_{i+1}, then at time t_{i+1}: $k^+ = k_{\texttt{arr}}$ and $p = p_{\texttt{arr}}$. If no task has arrived, then we say $k^+ = \perp$ and $p = 0$, i.e. $F_\perp(p) = H(p)$, where H is the Heaviside step function. Thus, k^+ equals the task class if a task has arrived since the last decision epoch, and equals $\perp$ if no task has arrived.

Note that we must include the combination of task classes currently being processed, $(n_1, \ldots, n_{|K|})$, in the state because each task class has a different service rate. Therefore, the rate at which tasks are being processed depends explicitly on the classes of tasks being processed. To simplify the notation, at times we will use the abbreviation $\mathbf{n} = (n_1, \ldots, n_{|K|}) \in \mathbf{N}$, where $\mathbf{N}$ is the set of possible task class combinations.

The action space is $A = \{\texttt{acc}, \texttt{rej}\}$, corresponding to accepting or rejecting a task. The $\texttt{acc}$ action can only be executed if $k^+ \neq \perp$ and $\sum_{k \in K} n_k < N_{serv}$, i.e. a task has arrived and there is at least one free server available. The reward for accepting a task is

$$R(\mathbf{x}, \texttt{acc}, t) = \mathbf{x}[p] \cdot \Pr(\texttt{fin}(\mathbf{x}[k^+]) \mid t), \quad (3)$$

where $\Pr(\texttt{fin}(k) \mid t) = 1 - \exp(-(t_H - t)\mu_k)$ is the probability that task class k will be completed before the horizon, t_H. Eq. 3 computes the expected price as the task price is only received if the task is completed before the horizon (Problem 1). The reward for rejecting a task is zero.

We now define the transition function for the $\texttt{rej}$ action. Due to independence between state variables, we can write the transition function in the following product form

$$P(\mathbf{x}' \mid \mathbf{x}, \texttt{rej}, t_i) = \Pr(\mathbf{x}'[k^+] \mid t_i) \times f_{\mathbf{x}'[k^+]}(\mathbf{x}'[p]) \times \prod_{k=1}^{K} \Pr(\mathbf{x}'[n_k] \mid \mathbf{x}[n_k]), \quad (4)$$

where the probability that no task arrives is

$$\Pr\left(\mathbf{x}'[k^+] = \perp \mid t_i\right) = \exp(-\Delta t \textstyle\sum_{k=1}^{K} \Lambda_k(t_i)), \quad (5)$$

and the probability that a task of class k' arrives is

$$\Pr(\mathbf{x}'[k^+] = k' \mid t_i) = \frac{\Lambda_{k'}(t_i)}{\sum_{k=1}^{K} \Lambda_k(t_i)} \left[1 - \Pr\left(\mathbf{x}'[k^+] = \perp \mid t_i\right)\right]. \quad (6)$$

The transition probabilities for n_k, the number of servers currently processing a task of class k, can be computed as

$$\Pr(\mathbf{x}'[n_k] \mid \mathbf{x}[n_k]) = \quad (7)$$
$$\begin{cases} \binom{\mathbf{x}[n_k]}{\mathbf{x}[n_k] - \mathbf{x}'[n_k]} \times (1 - e^{-\mu_k \Delta t})^{(\mathbf{x}[n_k] - \mathbf{x}'[n_k])} \times \\ \qquad e^{-\mu_k \mathbf{x}'[n_k] \Delta t}, \ \text{if } \mathbf{x}'[n_k] \leq \mathbf{x}[n_k]. \\ 0, \ \text{otherwise.} \end{cases}$$

where $\binom{\cdot}{\cdot}$ denotes the binomial coefficient. Eq. 7 follows a binomial distribution because each server finishing or not finishing during Δt is an independent Bernoulli trial.

We now consider the $\texttt{acc}$ action. If a task of class k is accepted, the reward $R(\mathbf{x}, \texttt{acc}, t_i)$ is received. The state instantaneously transitions to a successor state, $\mathbf{x}^{\texttt{acc}}_{(\mathbf{n},k)}$, where there is additional server processing task class k, and no task available to be accepted. Therefore, the Q-value for accepting a task can be computed by

$$Q_{\pi^*}(\mathbf{x}, \texttt{acc}, t_i) = \mathbf{x}[p] \cdot \Pr(\texttt{fin}(\mathbf{x}[k^+]) \mid t_i) + V_{\pi^*}(\mathbf{x}^{\texttt{acc}}_{(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+])}, t_i), \ \text{where} \quad (8)$$

$$\mathbf{x}^{\texttt{acc}}_{(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+])}[n_k] = \begin{cases} \mathbf{x}[n_k] + 1, \ \text{if } \mathbf{x}[k^+] = k \\ \mathbf{x}[n_k], \ \text{otherwise.} \end{cases} \quad (9)$$

and $\mathbf{x}^{\texttt{acc}}_{(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+])}[k^+] = \perp$. The subscript of $\mathbf{x}^{\texttt{acc}}_{(\mathbf{n},k)}$ indicates that it is the successor state after accepting a task of class k when the combination of tasks being processed was $\mathbf{n}$.

STA-HDMP Solution Algorithm

General solutions to HMDPs resort to function approximation due to the difficulty of optimising over the continuous state space (Kveton, Hauskrecht, and Guestrin 2006). However, as we shall show in this section, it is possible to solve the STA-HMDP exactly (i.e. without function approximation) using a finite number of Bellman backups. This is due to the specific structure of the optimal value function in the STA-HDMP with respect to the continuous state variable, p.

As a first step, we make the observation that the Q-value for rejecting the task only depends on $\mathbf{n}$ and t, i.e. $Q_{\pi^*}(\mathbf{x}, \texttt{rej}, t) = Q_{\pi^*}(\mathbf{x}', \texttt{rej}, t)$ if $\mathbf{x}[\mathbf{n}] = \mathbf{x}'[\mathbf{n}]$. Intuitively, this is because if a task is rejected, no immediate reward is received, and the transition dynamics to the next time step are the same irrespective of what task class and price was rejected. To make this explicit, we will write $Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \texttt{rej}, t)$ in place of $Q_{\pi^*}(\mathbf{x}, \texttt{rej}, t)$. Following from Eq. 2, the Q-value for the $\texttt{rej}$ action is

$$Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \texttt{rej}, t_i) = \quad (10)$$
$$\sum_{\mathbf{x}'[\mathbf{n}]} \sum_{\mathbf{x}'[k^+]} \int_{\mathbf{x}'[p]} P(\mathbf{x}' \mid \mathbf{x}, \texttt{rej}, t_i) \cdot V_{\pi^*}(\mathbf{x}', t_{i+1}) \cdot \mathrm{d}(\mathbf{x}'[p])$$

We now show that the optimal value function is piecewise-linear with respect to the task price, p. Full proofs of all propositions are in the supplementary material (supp. mat.) (Rigter et al. 2022).

Proposition 1. *Let $V_{\pi^*}(\mathbf{x}, t)$ be the optimal value function for the STA-HMDP. $V_{\pi^*}(\mathbf{x}, t)$ has the following form*

$$V_{\pi^*}(\mathbf{x}, t) = \begin{cases} \Pr\left(\mathtt{fin}(\mathbf{x}[k^+]) \mid t\right) \cdot \left(\mathbf{x}[p] - p^*_{cr}(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+], t)\right) \\ \qquad\qquad + Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \mathtt{rej}, t), \\ \qquad \text{if } \mathbf{x}[k^+] \neq \perp \text{ and } \mathbf{x}[p] \geq p^*_{cr}(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+], t). \\ Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \mathtt{rej}, t), \text{ otherwise.} \end{cases} \tag{11}$$

where p^*_{cr} is the *critical price function*, defined as

$$p^*_{cr}(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+], t) = \frac{1}{\Pr(\mathtt{fin}(\mathbf{x}[k^+]) \mid t)} \times \tag{12}$$
$$\Big(Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \mathtt{rej}, t) - Q_{\pi^*}(\mathbf{x}^{\mathtt{acc}}_{(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+])}[\mathbf{n}], \mathtt{rej}, t)\Big).$$

if $\mathbf{x}[k^+] \neq \perp$ and $t < t_H$. We define $p^*_{cr}(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+], t) = 0$ if $\mathbf{x}[k^+] = \perp$ or $t = t_H$.

Proposition 1 shows that the optimal value function is piecewise-linear with respect to $\mathbf{x}[p]$. For a given $\mathbf{x}[\mathbf{n}]$, $\mathbf{x}[k^+]$, and t, if we know $Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \mathtt{rej}, t)$ and the critical price function, we can compute the value function for any value of $\mathbf{x}[p]$. Additionally, the optimal policy is to accept tasks only if the task price exceeds the critical price.

Proposition 2. *[Optimal threshold policy] The optimal policy, π^*, for the STA-HMDP may be expressed as follows*

$$\pi^*(\mathbf{x}, t) = \begin{cases} \mathtt{acc}, \text{ if } \mathbf{x}[k^+] \neq \perp \text{ and } \mathbf{x}[p] \geq p^*_{cr}(\mathbf{x}[\mathbf{n}], \mathbf{x}[k^+], t) \\ \mathtt{rej}, \text{ otherwise} \end{cases}$$

Propositions 1 and 2 enable us to compute the optimal solution for all states using a finite number of Bellman backups. This is because to define the optimal value and policy for any state and time, with any continuous price, we only need to know $p^*_{cr}(\mathbf{n}, k, t)$ and $Q_{\pi^*}(\mathbf{n}, \mathtt{rej}, t)$ for all $\mathbf{n} \in \mathbf{N}, k \in K$, and $t \in D$. Therefore, there are only a finite number of Q-values and critical prices that we need to compute.

One remaining issue is that Eq. 10 contains integrals which would be expensive to compute at every Bellman backup. In Proposition 3 we show that the computations in Eq. 10 can be simplified in the following manner.

Proposition 3. *[Mean shortage function]*

$$Q_{\pi^*}(\mathbf{x}[\mathbf{n}], \mathtt{rej}, t_i) = \sum_{\mathbf{x}'[\mathbf{n}]} \prod_{k=1}^{K} \Pr(\mathbf{x}'[n_k] \mid \mathbf{x}[n_k]) \cdot \tag{13}$$
$$\sum_{\mathbf{x}'[k^+]} \Pr(\mathbf{x}'[k^+] = k^{+\prime} \mid t_i) \cdot \Big[Q_{\pi^*}(\mathbf{x}'[\mathbf{n}], \mathtt{rej}, t_{i+1}) +$$
$$\Pr\left(\mathtt{fin}(\mathbf{x}'[k^+]) \mid t_{i+1}\right) \cdot \phi_{\mathbf{x}'[k^+]}\big(p^*_{cr}(\mathbf{x}'[\mathbf{n}], \mathbf{x}'[k^+], t_{i+1})\big)\Big]$$

where $\phi_{k^+}(p)$ is the *mean shortage function* of the price distribution for task class k^+ defined as $\phi_{k^+}(p) = \int_p^\infty (1 - F_{k^+}(y))\,\mathrm{d}y$.

Proposition 3 means that we do not have to compute the integrals in Eq. 10 separately for each Bellman backup, but instead we can query the mean shortage function. The mean shortage function can be computed in closed form for many common distributions (e.g. exponential, Pareto), and can be computed efficiently for the nonparametric representation of the task price distribution used in Dervovic et al. (2021).

We are now ready to introduce our finite horizon value iteration algorithm for the STA-HMDP, with pseudocode in Alg. 1, supp. mat. We initialise the value function to zero at the horizon, t_H. We then iterate over each time step backwards in time. For a given time step, we 1) compute the Q-value corresponding to the $\mathtt{rej}$ action using Eq. 10 and 13, and 2) compute the critical price values using Eq. 12. After we have computed the critical price function for all time steps, we can derive the optimal policy using Proposition 2.

STA-HDMP Approximation Error

We now establish that as the time step size, Δt, approaches zero the optimal solution to the STA-HMDP approaches the optimal continuous time solution to Problem 1.

Proposition 4. *Let $\overline{V}^*_\pi(\mathbf{x}, t)$, and $\overline{p}^*_{cr}(\mathbf{n}, k^+, t)$ be the optimal value function and critical price function for the continuous time problem defined by Problem 1. Let $V_{\pi^*}(\mathbf{x}, t_i)$, and $p^*_{cr}(\mathbf{n}, k^+, t_i)$ be the optimal value function and critical price function for the corresponding STA-HMDP at decision epochs $t_i \in D$. Then:*

$$\lim_{\Delta t \to 0^+} \mid \overline{V}_{\pi^*}(\mathbf{x}, t_i) - V_{\pi^*}(\mathbf{x}, t_i) \mid = 0, \ \forall\, \mathbf{x},\ t_i \in D$$

$$\lim_{\Delta t \to 0^+} \mid \overline{p}^*_{cr}(\mathbf{n}, k^+, t_i) - p^*_{cr}(\mathbf{n}, k^+, t_i) \mid = 0, \ \forall\, \mathbf{n},\ k^+,\ t_i \in D$$

Approximation via State Abstraction

In the previous section, we have shown how to solve the STA-HDMP. However, this requires iterating over the set $\mathbf{N}$ of possible combinations of tasks currently being processed. The size of this set grows rapidly with the number of task classes, prohibiting scalability to more than a few classes. Intuitively, we expect that some combinations of tasks result in a similar "workload". For example, processing two tasks with a medium service rate might represent a similar workload to processing one fast, and one slow task. If this were the case, we would expect that the optimal policy would apply a similar threshold to accepting new tasks in both cases. Therefore, we can expect to obtain a good solution by treating both of these cases as the same. This idea of grouping similar states together is referred to as *state abstraction* (Li, Walsh, and Littman 2006). In this section, we show how to use state abstraction to improve the scalability of our approach.

We introduce the following notation following from Li, Walsh, and Littman (2006). Let $\mathcal{M} = (\mathbf{X}, A, R, P, D)$ be referred to as the *ground* MDP, with optimal policy π^*. The *abstract* version of $\mathcal{M}$ is $\widehat{\mathcal{M}} = (\widehat{\mathbf{X}}, A, \widehat{R}, \widehat{P}, D)$, and has optimal policy $\widehat{\pi}^*$. The abstraction function, $\psi : \mathbf{X} \to \widehat{\mathbf{X}}$, maps each ground state to its corresponding abstract state, and $\psi^{-1}(\widehat{\mathbf{x}})$ is the inverse of $\psi(\mathbf{x})$. The *weighting function* is $w : \mathbf{X} \to [0, 1]$, where for each $\widehat{\mathbf{x}} \in \widehat{\mathbf{X}}$, $\sum_{\mathbf{x} \in \psi^{-1}(\widehat{\mathbf{x}})} w(\mathbf{x}) = 1$. The abstract reward and transition functions are a weighted sum over the corresponding functions for the ground states

$$\widehat{R}(\widehat{\mathbf{x}}, a, t) = \sum_{\mathbf{x} \in \psi^{-1}(\widehat{\mathbf{x}})} w(\mathbf{x}) R(\mathbf{x}, a, t) \tag{14}$$

$$\widehat{P}(\widehat{\mathbf{x}}' \mid \widehat{\mathbf{x}}, a, t) = \sum_{\mathbf{x} \in \psi^{-1}(\widehat{\mathbf{x}})} \sum_{\mathbf{x}' \in \psi^{-1}(\widehat{\mathbf{x}}')} w(\mathbf{x}) P(\mathbf{x}' \mid \mathbf{x}, a, t) \tag{15}$$

To generate the state aggregation function, one possibility is to consider aggregating states together when their optimal Q-values in the ground MDP are within ε, i.e.

$$\psi(\mathbf{x}_1) = \psi(\mathbf{x}_2) \to \forall_{a \in A} \mid Q_{\pi^*}(\mathbf{x}_1, a, t) - Q_{\pi^*}(\mathbf{x}_2, a, t) \mid \leq \varepsilon \tag{16}$$

Let π^{GA} denote the policy in the ground MDP derived from the optimal policy in the abstract MDP,

$$\pi^{GA}(\mathbf{x}) = \widehat{\pi}^*\big(\psi(\mathbf{x})\big). \tag{17}$$

Abel, Hershkowitz, and Littman (2016) prove that for any valid weighting function, if the state aggregation function satisfies Eq. 16, then the suboptimality of π^{GA} in the ground MDP is bounded by a function linear in ε:

$$V_{\pi^*}(\mathbf{x}) - V_{\pi^{GA}}(\mathbf{x}) \leq \mathcal{O}(\varepsilon), \text{ for all } \mathbf{x} \in \mathbf{X} \tag{18}$$

State Abstraction in the STA-HDMP

We are now ready to present our state abstraction approach for approximating the optimal solution to the STA-HMDP.

Proposition 5. *Consider two states, $\mathbf{x}_1$ and $\mathbf{x}_2$, in the STA-HDMP where $\mathbf{x}_1[k^+] = \mathbf{x}_2[k^+]$ and $\mathbf{x}_1[p] = \mathbf{x}_2[p]$. If $\mid Q_{\pi^*}(\mathbf{x}_1[\mathbf{n}], \texttt{rej}, t) - Q_{\pi^*}(\mathbf{x}_2[\mathbf{n}], \texttt{rej}, t) \mid \leq \varepsilon/2$, and $\forall_{k^+} \mid p^*_{cr}(\mathbf{x}_1[\mathbf{n}], k^+, t) - p^*_{cr}(\mathbf{x}_2[\mathbf{n}], k^+, t) \mid \leq \varepsilon/2$, then*

$$\forall_{a \in A} \mid Q_{\pi^*}(\mathbf{x}_1, a, t) - Q_{\pi^*}(\mathbf{x}_2, a, t) \mid \leq \varepsilon \tag{19}$$

Proposition 5 suggests that we can define the state aggregation function as follows

$$\begin{aligned}\psi(\mathbf{x}_1) = \psi(\mathbf{x}_2) \to \mathbf{x}_1[k^+] = \mathbf{x}_2[k^+],\ \mathbf{x}_1[p] = \mathbf{x}_2[p],\\ Q_{\pi^*}(\mathbf{x}_1[\mathbf{n}], \texttt{rej}, t) \approx Q_{\pi^*}(\mathbf{x}_2[\mathbf{n}], \texttt{rej}, t) \text{ and}\\ p^*_{cr}(\mathbf{x}_1[\mathbf{n}], k^+, t) \approx p^*_{cr}(\mathbf{x}_2[\mathbf{n}], k^+, t) \text{ for all } k^+,\end{aligned} \tag{20}$$

and the performance of the policy derived from the abstract MDP will be near-optimal due to the result stated in Eq. 16-18. The resulting abstract states are represented by the state variables $\{n_A, k^+, p\}$, where $n_A \in \mathcal{N}_A$ is an abstraction of $\mathbf{n}$, the combination of task classes being processed in the ground state. $\mathcal{N}_A$ is the set of abstractions of task class combinations. We write $\psi_{\mathbf{n}} : \mathbf{N} \to \mathcal{N}_A$ to denote the aggregation function for combinations of task classes such that

$$\begin{aligned}\psi(\mathbf{x}_1) = \psi(\mathbf{x}_2) \Longleftrightarrow \mathbf{x}_1[k^+] = \mathbf{x}_2[k^+],\ \mathbf{x}_1[p] = \mathbf{x}_2[p],\\ \text{and } \psi_{\mathbf{n}}(\mathbf{x}_1[\mathbf{n}]) = \psi_{\mathbf{n}}(\mathbf{x}_2[\mathbf{n}])\end{aligned} \tag{21}$$

We cannot directly apply Proposition 5 to aggregate task combinations since we cannot compute the optimal STA-HDMP values required from the scaling of Alg. 1 for many task classes. Guided by the intuition obtained from Eq. 20 we propose two more scalable methods for computing the state aggregation function.

State Aggregation via Semi-MDP Stationary Solution We propose to compute a stationary approximation of the STA-HMDP solution to determine the aggregation function. First, we compute the average arrival rate over the horizon for each task class, $\mathbb{E}_t[\Lambda_k(t)]$. We then find the solution which optimises the average reward over an infinite horizon using the average arrival rates. This problem can be solved more quickly than the STA-HDMP as the optimal solution is stationary. We solve this stationary problem by adapting policy iteration for stationary Semi-MDPs (Puterman 2005). Details of this are in the supp. mat. For each $\mathbf{n} \in \mathbf{N}$, we then compose a vector $v_{\mathbf{n}}$ of the associated *relative*[1] Q-value and critical price values in the stationary solution: $v^{ss}_{\mathbf{n}} = [\widetilde{Q}_{\pi^*}(\mathbf{n}, \texttt{rej}),\ \widetilde{p}^*_{cr}(\mathbf{n}, k_1), \ldots,\ \widetilde{p}^*_{cr}(\mathbf{n}, k_{|K|})]$, where we use "$\sim$" to denote the solution for the stationary problem. We then perform clustering on the $v^{ss}_{\mathbf{n}}$ vectors to aggregate the task class combinations into the desired number of abstractions, $|\mathcal{N}_A|$. These clusters define aggregation function, $\psi_{\mathbf{n}}$. For clustering, we use the k-means algorithm.

State Aggregation via Order Statistics The scalability of the state aggregation approach we have just outlined is limited, as it requires solving for the stationary solution which may not be feasible for a large number of task classes. Here, we present an alternative state aggregation approach, based on summary statistics for each combination of tasks. For each $\mathbf{n} \in \mathbf{N}$, we compose a vector of the form

$$v^{os}_{\mathbf{n}} = \big[\mathbb{E}[t_{N_f \geq 0} \mid \mathbf{n}], \mathbb{E}[t_{N_f \geq 1} \mid \mathbf{n}], \ldots, \mathbb{E}[t_{N_f = N_{serv}} \mid \mathbf{n}]\big]$$

where N_f denotes the number of servers free, i.e. $N_f = N_{serv} - \sum_{k \in K} n_k$. Additionally, $\mathbb{E}[t_{N_f \geq q} \mid \mathbf{n}]$ denotes the expected time until at least q servers are free given that the combination of tasks currently being processed is $\mathbf{n}$, and no further tasks are accepted. The intuition for $v^{os}_{\mathbf{n}}$ is that it approximately summarises the distribution over task completion times for the tasks currently being processed. Once the vectors have been computed, we perform clustering using k-means, and the resulting clusters correspond to $\psi_{\mathbf{n}}$.

Computing $v^{os}_{\mathbf{n}}$ requires computing the mean of order statistics of independent and non-identical exponential distributions. This is computationally challenging, so in practice we use a Monte Carlo approximation (details in supp. mat.).

Value Iteration for STA-HMDP with State Abstraction We refer to the abstract version of the STA-HDMP as the Abstract STA-HDMP. We provide a brief summary of the algorithm for the Abstract STA-HDMP here, and more details can be found in the supp. mat. We denote by $\widehat{Q}_{\pi^*}$, and $\widehat{p}^*_{cr}$ the optimal Q-value function and critical price respectively in the Abstract STA-HDMP. Like the original STA-HMDP, the optimal policy for the Abstract STA-HMDP has a threshold form, as stated in Proposition 6.

Proposition 6. *The optimal policy for the Abstract STA-HMDP, $\widehat{\pi}^*$, may be expressed as follows*

$$\widehat{\pi}^*(\widehat{\mathbf{x}}, t) = \begin{cases} \texttt{acc}, & \text{if } \widehat{\mathbf{x}}[k^+] \neq \perp \text{ and } \widehat{\mathbf{x}}[p] \geq \widehat{p}^*_{cr}(\widehat{\mathbf{x}}[n_A], \widehat{\mathbf{x}}[k^+], t) \\ \texttt{rej}, & \text{otherwise} \end{cases}$$

[1]We use the relative value (see Puterman (2005), Chapter 8) as the value is infinite the average-reward infinite horizon setting.

where the critical price function is

$$\widehat{p}^*_{cr}(n_A, k^+, t) = \frac{1}{\Pr(\mathtt{fin}(k^+) \mid t)} \times \Big(\widehat{Q}_{\pi^*}(n_A, \mathtt{rej}, t) - \sum_{\mathbf{n} \in \psi_{\mathbf{n}}^{-1}(n_A)} w(\mathbf{n}) \cdot \widehat{Q}_{\pi^*}(\psi(\mathbf{x}^{\mathtt{acc}}_{(\mathbf{n},k^+)}), \mathtt{rej}, t)\Big). \quad (22)$$

if $\mathbf{x}[k^+] \neq \perp$ *and* $t < t_H$*, and 0 otherwise.*

The value iteration algorithm for solving the Abstract STA-HDMP proceeds in a similar manner to the STA-HMDP. Pseudocode is provided in Alg. 2 in the supp. mat. Once the policy has been computed for the Abstract STA-HDMP, the policy to apply to the original STA-HMDP is derived using Eq. 17.

Experiments

The experiments were run using an Intel Xeon E3-1585L v5 3GHz processor with 64GB of RAM. We compare several methods based on our Value Iteration (VI) approach as well as additional baselines. The following methods are compared:

- *VI No Abstr.*: our VI approach from Alg. 1.
- *VI Stationary Sol. Abstr.*: our VI approach with state abstraction based on the Semi-MDP stationary solution.
- *VI Order Stat. Abstr.*: our VI approach with state abstraction based on the mean of order statistics.
- *VI Random Abstr.*: our VI approach with state abstraction with random state aggregation.
- *VI Avg. Class*: we compute an "average" task class, k_{avg}, as follows. The arrival rate function is $\Lambda_{k_{\text{avg}}}(t) = \sum_{k \in K} \Lambda_k(t)$. The price distribution and service rate are a weighted average over each task. The weighting for each class, ω_k, is proportional to the mean arrival rate for each class: $\omega_k \propto \mathbb{E}_t[\Lambda_k(t)]$. We perform VI using the single average class, and assume that all tasks are the average class.
- *Stationary Sol.*: uses the critical prices from the stationary solution, which assumes constant arrival rates.
- *Grid Search*: the critical price for each task class, p_k, is proportional to the expected service time, i.e. $p_k = C/\mu_k$, where C is a constant. To find the best value for C, we evaluate the performance of 50 log-spaced values of C for 300 episodes each, and choose the best value.

Synthetic Domains For each synthetic domain, the horizon is 28800s, representing an 8hr working day. The task prices are distributed according to a Lomax distribution, with shape parameter of 3. We test two arrival rate functions: *sinusoid* and *step* functions which are plotted in the supp. mat.
Small: There are 10 servers and 3 task classes which we call $\{\mathtt{slow}, \mathtt{medium}, \mathtt{fast}\}$. The scale parameters of the Lomax price distribution for each task class are: $\{\mathtt{slow}{=}1600, \mathtt{medium}{=}900, \mathtt{fast}{=}400\}$. The service rates for each task class are: $\{\mathtt{slow}{=}\frac{1}{2000}, \mathtt{medium}{=}\frac{1}{1000}, \mathtt{fast}{=}\frac{1}{500}\}$.
Large: There are 40 servers and 4 task classes which we call $\{\mathtt{very\,slow}, \mathtt{slow}, \mathtt{medium}, \mathtt{fast}\}$. The Lomax price distribution scale parameters are: $\{\mathtt{very\,slow}{=}2500, \mathtt{slow}{=}1600, \mathtt{medium}{=}900, \mathtt{fast}{=}400\}$. The service rates for each of the task classes are: $\{\mathtt{very\,slow}{=}\frac{1}{4500}, \mathtt{slow}{=}\frac{1}{3000}, \mathtt{medium}{=}\frac{1}{1500}, \mathtt{fast}{=}\frac{1}{750}\}$.

For both versions of the synthetic domains, tasks with slower service times tend to have higher prices.

Public Fraud Data For this domain we use a public dataset, M, of financial transactions which are labelled as fraudulent or non-fraudulent (IEEE-CIS 2019). We wish to optimise the expected value of fraudulent transactions reviewed by staff. The horizon for each episode is 86400s (1 day). We use half of the dataset, M_{train}, to train a classifier to predict the probability that a transaction is fraudulent based on additional features of the data. We assume that transactions with a probability ≥ 0.1 are automatically reviewed. Our objective is to optimise the value of the remaining transactions which are reviewed by staff. To adjust for the fraud likelihood, we consider the *adjusted price* of each task (transaction) to be the value of the transaction multiplied by the probability of fraud predicted by the classifier. We assume that tasks are divided into the following classes according to the value of the transactions: $\{\mathtt{low\_val}\text{: } [\$0, \$150),\ \mathtt{med\_val}\text{: } [\$150, \$500),\ \mathtt{high\_val}\text{: } [\$500, \infty)\}$. We assume that there are 20 staff (servers), and that transactions of different values require different processes to review, and therefore have the following service rates: $\{\mathtt{low\_val}{=}\frac{1}{900}, \mathtt{med\_val}{=}\frac{1}{3600}, \mathtt{high\_val}{=}\frac{1}{10{,}800}\}$. We learn the adjusted price distributions and arrival rate functions from M_{train} with the method from Dervovic et al. (2021). We then simulate the total adjusted price of transactions reviewed using the remainder of the data, M_{test}.

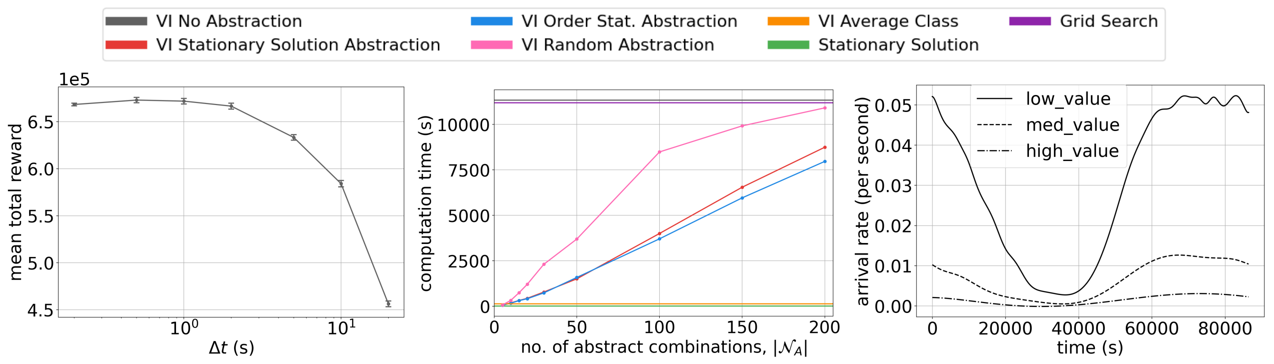
Results

We evaluate each approach for 300 episodes on each domain. In the plots, the error bars and the shaded regions indicate the mean $\pm$ the std. error. Additional results are in the supp. mat.

Influence of time step Figure 1a shows the influence of Δt for the *VI No Abstr.* method in the Synthetic Small domain with sinusoid arrival rate functions. At large values of Δt, the performance is poor, but as Δt is decreased below approximately 1 second, the performance plateaus. This empirically confirms Proposition 4. Based on this finding, we use a time step of $\Delta t = 0.5$s for all other results on the synthetic domains. For the fraud domain we use $\Delta t = 1$s as this is the resolution of the time information in the dataset.

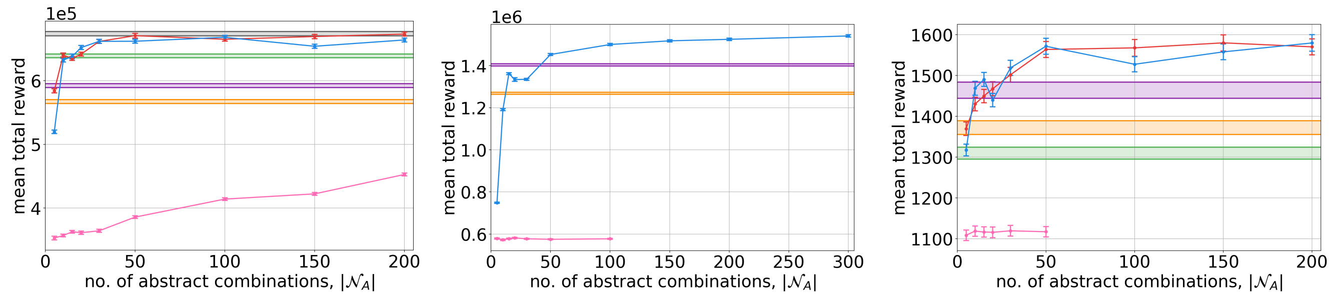
Computation Time The computation times for each approach on the Synthetic Small domain are shown in Figure 1b. *VI No Abstr.* and *Grid Search* are the slowest methods. For the methods based on state abstraction, the computation time decreases as the number of abstractions is decreased, verifying that our abstraction method reduces the computation required. We omit results for methods which take more than 30,000s, including *VI No Abstr.* on either of the larger domains. Computation times for the other domains are included in the supp. mat.

Synthetic Domain Results Results on the Synthetic Small domain with sinusoid arrival rates are in Figure 2a. *VI No Abstr.* has the strongest performance, and the best baseline is *Stationary Sol.* Both *VI Order Stat. Abstr.* and *VI Stationary Sol. Abstr.* perform better than all baselines, and comparably



(a) Influence of time step. (b) Computation times. (c) Arrival rate functions.

Figure 1: Mean reward vs Δt for *VI No Abstractions (a)*, computation time for each method *(b)* in Synthetic Small domain (sinusoid arrival rates), and *(c)* Public fraud dataset arrival rates.



(a) Synthetic Small (sinusoid arrival rates). (b) Synthetic Large (sinusoid arrival rates). (c) Public Fraud dataset.

Figure 2: Mean total reward performance on 300 evaluation runs for each domain. Error bars and shaded regions indicate standard errors. We only include methods and numbers of abstract combinations which can be computed within 30,000 seconds.

to *VI No Abstr.* if a sufficient number of abstractions ($\geq$ 30) are used. *VI Random Abstr.* performs poorly across all domains, indicating that the state aggregation approaches we have proposed are crucial for strong performance.

The results on the Synthetic Large domain in Figure 2b show that *VI Order Stat. Abstr.* is the best performing method, provided that a sufficient number of state abstractions are used. For the large domain, the number of possible combinations of task classes is over 130,000. This means that we cannot provide results for *VI No Abstr.*, or any of the methods which require the stationary solution because these methods (considerably) exceed our 30,000s computation limit.

In the supp. mat., we include results for the synthetic domains using step functions for the arrival rates. We observe similar results for the step functions, indicating that our method performs well for a variety of arrival rate functions.

Public Fraud Data Results For the fraud domain, the arrival rates are plotted in Figure 1c, and the performance of each method is in Figure 2c. Our methods based on state abstraction outperform all of the baselines when using $\geq$ 30 state abstractions, and require less computation time than the grid search approach as shown in the supp. mat.

Our results validate that a) our state abstraction approach reduces the computation requirements for value iteration, making it applicable to complex problems, and b) our approach significantly improves performance over simpler baselines, and therefore has the potential to improve the efficiency of real world systems, such as those used to monitor fraud.

Conclusion

We have introduced a novel queuing problem that corresponds to a challenge faced in financial fraud detection and monitoring. Moreover, we demonstrate that it has a computationally tractable solution by discretising time and utilising state abstraction. Future work includes adversarial modelling of incoming tasks and proving reward lower bounds as a function of the size of the abstract state space.

References

Abel, D.; Hershkowitz, D.; and Littman, M. 2016. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, 2915–2923. PMLR.

Albright, S. C. 1974. Optimal sequential assignments with random arrival times. *Management Science*, 21(1): 60–67.

Ata, B. 2006. Dynamic control of a multiclass queue with thin arrival streams. *Operations Research*, 54(5): 876–892.

Ata, B.; and Tongarlak, M. H. 2013. On scheduling a multiclass queue with abandonments under general delay costs. *Queueing Systems*, 74(1): 65–104.

Bellman, R. 1966. Dynamic programming. *Science*, 153(3731): 34–37.

Bertsimas, D.; Paschalidis, I. C.; and Tsitsiklis, J. N. 1994. Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. *The Annals of Applied Probability*, 43–75.

Cao, P.; and Xie, J. 2016. Optimal control of a multiclass queueing system when customers can change types. *Queueing Systems*, 82(3-4): 285–313.

Dal Pozzolo, A.; Caelen, O.; Le Borgne, Y.-A.; Waterschoot, S.; and Bontempi, G. 2014. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10): 4915–4928.

Derman, C.; Lieberman, G. J.; and Ross, S. M. 1972. A sequential stochastic assignment problem. *Management Science*, 18(7): 349–355.

Dervovic, D.; Hassanzadeh, P.; Assefa, S.; and Reddy, P. 2021. Non-Parametric Stochastic Sequential Assignment With Random Arrival Times. *International Joint Conference on Artificial Intelligence*.

Duckworth, P.; Lacerda, B.; and Hawes, N. 2021. Time-Bounded Mission Planning in Time-Varying Domains with Semi-MDPs and Gaussian Processes. *Conference on Robot Learning*.

Ghosh, M. K.; and Saha, S. 2013. Non-stationary semi-Markov decision processes on a finite horizon. *Stochastic Analysis and Applications*, 31(1): 183–190.

Harrison, J. M.; and Zeevi, A. 2004. Dynamic scheduling of a multiclass queue in the Halfin-Whitt heavy traffic regime. *Operations Research*, 52(2): 243–257.

Howard, R. A. 1963. Semi-markovian decision-processes. *Proceedings of the International Statistical Institute*, 40(2): 625–652.

IEEE-CIS. 2019. IEEE Computational Intelligence Society Fraud Detection. https://www.kaggle.com/c/ieee-fraud-detection. Accessed: 2021-08-28.

Jewell, W. S. 1963. Markov-renewal programming. I: Formulation, finite return models. *Operations Research*, 11(6): 938–948.

Kleywegt, A. J.; and Papastavrou, J. D. 1998. The dynamic and stochastic knapsack problem. *Operations research*, 46(1): 17–35.

Klimov, G. 1974. Time-sharing service systems. I. *Theory of Probability & Its Applications*, 19(3): 532–551.

Kveton, B.; Hauskrecht, M.; and Guestrin, C. 2006. Solving factored MDPs with hybrid state and action variables. *Journal of Artificial Intelligence Research*, 27: 153–201.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a Unified Theory of State Abstraction for MDPs. *ISAIM*, 4: 5.

Mamer, J. W. 1986. Successive approximations for finite horizon, semi-Markov decision processes with application to asset liquidation. *Operations Research*, 34(4): 638–644.

McMahon, J. J. 2008. *Time-dependence in Markovian decision processes*. Ph.D. thesis, University of Adelaide, Australia.

Puterman, M. L. 2005. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Rigter, M.; Dervovic, D.; Hassanzadeh, P.; Long, J.; Zehtabi, P.; and Magazzeni, D. 2022. Optimal Admission Control for Multiclass Queues with Time-Varying Arrival Rates via State Abstraction. arXiv:2203.08019.

Rue, R. C.; and Rosenshine, M. 1985. The application of semi-Markov decision processes to queueing of aircraft for landing at an airport. *Transportation science*, 19(2): 154–172.

Sennott, L. I. 1989. Average cost semi-Markov decision processes and the control of queueing systems. *Probability in the Engineering and Informational Sciences*, 3(2): 247–272.

Stidham, S. 1985. Optimal control of admission to a queueing system. *IEEE Transactions on Automatic Control*, 30(8): 705–713.

Yoon, S.; and Lewis, M. E. 2004. Optimal pricing and admission control in a queueing system with periodically varying parameters. *Queueing Systems*, 47(3): 177–199.