Scheduling for Urban Air Mobility using Safe Learning

Surya Murthy
School of Computer Engineering
University of Illinois, Urbana-Champaign
Urbana, Illinois
suryakm2@illinois.edu

Natasha A. Neogi
NASA Langley Research Center
Hampton, Virginia
natasha.a.neogi@nasa.gov

Suda Bharadwaj
Skygrid LLC.
Austin, Texas
sbharadwaj@skygrid.com

Abstract. This work considers the scheduling problem for Urban Air Mobility (UAM) vehicles travelling between origin-destination pairs with both hard and soft trip deadlines. Each route is described by a discrete probability distribution over trip completion times (or delay) and over inter-arrival times of requests (or demand) for the route along with a fixed hard or soft deadline. Soft deadlines carry a cost that is incurred when the deadline is missed. An online, safe scheduler is developed that ensures that hard deadlines are never missed and that the average cost of missing soft deadlines is minimized. The system is modelled as a Markov Decision Process (MDP) and safe model based learning is used to find the probabilistic distributions over route delays and demand. Monte Carlo Tree Search (MCTS) Earliest Deadline First (EDF) is used to safely explore the learned models in an online fashion and develop a near-optimal non-preemptive scheduling policy. These results are compared with Value Iteration (VI) and MCTS (Random) scheduling solutions.

1 Introduction

The development of emerging aviation markets is crucial to maintaining international competitiveness and providing benefit to the US economy and general public. The safe development of an air transportation system that enables passengers, cargo, and payloads to execute their mission via aerial means has been a major driver in the past and current century. Urban Air Mobility (UAM) refers to the transportation of passengers and cargo over an urban environment, and it is often assumed to occur in an on-demand (aperiodic) manner. UAM aims to alleviate urban congestion and improve urban public transit. UAM vehicles are assumed to take off and land at a series of localized vertiports, vertihubs, or vertistations. Within the environment, the vehicles receive trip requests to move from one vertihub to another. By correctly scheduling these trip requests, the vehicles can navigate the UAM airspace safely and efficiently. Thus, the development of a safe scheduler (i.e., no hard request misses its deadline and all hard requests are serviced) which does not require clairvoyance (i.e., scheduler does not have knowledge of the characteristics of the request, such as trip execution time or request inter-arrival time) is critical. A key question then becomes: **Given a set of origin-destination trip requests, where the requested time of arrival for each trip may either be a hard or soft deadline, can a schedule be created that guarantees all trips with hard deadlines arrive within their deadline and that the average cost of delayed trips with soft deadlines is minimized?**

1.1 Contribution

The major contributions of this work are: (1) The infinite duration, aperiodic, non-preemptive scheduling problem for UAM trip requests with both hard and soft deadlines is formally modelled as a non-standard optimization problem over a Markov Decision Process (MDP). (2) The demand and delay distributions of the UAM trip requests (which are not known a priori) are safely learned over the MDP model using

Submitted to:
FMAS 2022

sampling techniques. (3) Monte Carlo Tree Search (MCTS) is used to learn safe, scalable non-preemptive scheduling strategies over the learned models. (4) Results are compared to solutions obtained using value iteration and Earliest Deadline First (EDF) scheduling techniques. The developed approach scales well for a small number of UAM trip routes where each route can have infinitely many trip requests.

1.2 Related Work

The scheduling of UAM vehicles is a relatively new topic and has been considered from an operational concept point of view [22]. There have been some initial forays into building schedules for heterogeneous fleets of vehicles which focus on efficiency [13]. Heuristic approaches for arrival sequencing have been put forth [20] as have energy efficient approaches [14]. A graph based reinforcement learning approach has been employed in [19], where the UAM fleet scheduling problem is represented as an MDP over a graph space, with the graph representing the network of vertiports (and their dynamic properties, e.g., demand). However, none of these approaches enforce safety guarantees over the synthesized schedule (i.e., all hard trip requests are completed on schedule).

The classical real-time scheduling problem has been explored in great detail [9],[3],[5],[7],[8]. The scheduling of periodic tasks with hard real-time deadlines has been shown to be co-NP-complete [16],[2], and there has been considerable study of the case where the tasks are not strictly periodic [12],[15]. The scheduling of systems which have both hard and soft deadlines has also been investigated [1],[17]. Classical algorithms for scheduling both hard and soft tasks in a preemptible fashion include Earliest Deadline First [18], Earliest Deadline Late [6], and Priority Exchange [23] among others. These techniques do not consider probabilistic distributions across task completion times and inter-arrival times.

In [10], the stochastic scheduling problem with both hard and soft tasks on a single machine is considered, and a safe and efficient non-clairvoyant scheduler is computed in an online fashion. The system is modelled as a finite MDP, and an algorithm for safe and optimal scheduler synthesis is developed. In a follow-on effort by the authors contained in [4], safe learning for near-optimal scheduling is examined. Monte Carlo Tree search techniques are leveraged to learn safe strategies for scheduling systems with large state spaces (e.g., 10^{20}). However, neither of these efforts consider non-preemptible tasks.

2 UAM Scheduling Problem

The UAM scheduling problem is modeled as an MDP. An MDP is defined as the tuple (S, A, δ, R) where: (1) S is a finite set of states; (2) A is a finite set of actions; (3) $\delta(s, a, s')$ is the probability that system goes from state s to state s' after action a is taken; and (4) R is the reward obtained immediately after action a is taken (with $|R| \leq B \in \mathbb{R}$ being a finite bounded reward chosen a priori). Informally, the states of the MDP are the possible trip request configurations available within the system at the current time step. The actions are the requests the scheduler can work on at that time step (i.e., the trips available to execute given that a UAM vehicle is available for that particular trip). The action space represents the actions the scheduler can take at any given state (i.e., the set of all possible trips that can be scheduled given UAM fleet availability).

2.1 UAM Scheduling Term Definitions

Definition 2.1. Trip completion time probability distribution (p). Let T be a discrete random variable drawn from a categorical distribution. The possible trip completion times are found in the support vector of the distribution denoted as $\mathscr{E}_T = \{0, 1, 2, ..., K\}$ where $K \in \mathbb{Z}^{\text{nonneg}}$ is the maximum possible

trip completion time. Define the trip completion time probability distribution $\mathbf{p} = \{p_0, \ldots, p_K\}$ where p_i corresponds to the *probability of the trip completion time T being equal to i.*

Example 1. *A trip completion time probability distribution of* $\mathbf{p} = \{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}$ *has a maximum completion time* $K = 4$ *and a support vector of* $\mathscr{E}_T = \{0,1,2,3,4\}$. *Informally, this distribution means that the trip will complete in 3 time steps ($T = 3$) or 4 time steps ($T = 4$) with equal probability (0.5).*

Definition 2.2. Inter-arrival time probability distribution (q). Define the inter-arrival time probability distribution $q = \{q_0, \ldots, q_M\}$ with support vector $\mathscr{E}_V = \{0,1,2,...,M\}$ where $M \in \mathbb{Z}^{\text{nonneg}}$ is the maximum possible inter-arrival time and the i^{th} component of q corresponds to the *probability of the inter-arrival time V being equal to i.*

Example 2. *An inter-arrival time distribution of* $q = \{q_0 = 0, q_1 = 0, ..., q_8 = 1\}$ *has a maximum inter-arrival time* $M = 8$*, and a support vector of* $\mathscr{E}_V = \{0,1,2,...,8\}$. *Informally, this distribution means that a new trip request for that origin-destination pair will arrive in 8 time steps ($V = 8$) with probability 1.0.*

Definition 2.3. Deadline (D). Define D as a positive integer that denotes the number of time steps by which a scheduled trip request *must* be completed. In order to guarantee a trip can be scheduled without violating the deadline, assume $K \leq D \leq M$.

Example 3. *A deadline of 7 is denoted by* $D = 7$. *Informally, this means that the trip request must be completed within 7 time steps of being started.*

Definition 2.4. Route. A *route* is a template for all trip requests of a given type. Define the set ROUTES as the set of all origin-destination vertiport pairs and denote its cardinality as W.

Definition 2.5. Request. A *request* is a dynamic instantiation of a route. The following definition is presented for the case where there can only exist one request from a given route in the system at a time, without loss of generality. Formally,

$$r_i = (\mathbf{p_i}, D_i, \mathbf{q_i}) = (\{p_{i_0}, p_{i_1}, p_{i_2}, ..., p_{i_K}\}, D_i, \{q_{i_0}, q_{i_1}, ..., q_{i_M}\}) : \forall\, i \in \text{ROUTES}.$$

If there is no ambiguity, the subscripts in the tuples p_i and q_i may be dropped (e.g., $r_i = (\{p_0, p_1, p_2, ..., p_K\}$, $D_i, \{q_0, q_1, ..., q_M\}$). Requests have an initial configuration. When a new instance of a request arrives, the request will be initialized in the initial configuration $r_{i_{in}} = (\mathbf{p_{i_{in}}}, D_{i_{in}}, \mathbf{q_{i_{in}}})$. Also, the initial trip completion time, initial deadline, and initial inter-arrival time cannot be equal to zero (e.g., $\mathbf{p_{i_{in}}}, \mathbf{q_{i_{in}}} \neq \{1,0,...,0\}, D_{i_{in}} \neq 0$).

Example 4. *Consider an MDP with 2 requests as seen in Figure 1. The initial configuration of the first request is* $(\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}, 7, \{q_0 = 0, q_1 = 0, ..., q_8 = 1\})$. *This request has a completion time distribution of* $\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}$*, a deadline of 7, and an inter-arrival time of* $\{q_0 = 0, q_1 = 0, ..., q_8 = 1\}$. *Informally, it means that this request will take 3 or 4 time steps of dedicated work to complete, must be completed within 7 time steps, and will be replaced by a new request in 8 time steps. These values can change as time moves forward.*

Definition 2.6. Soft requests. The set of soft requests, r_{soft}, is defined as all of the requests that can be completed after their deadline reaches 0. However, a cost (or negative reward R) will be incurred for missing the deadline for that request. Formally, $r_{soft} = \{(\mathbf{p}_0, D_0, \mathbf{q}_0), (\mathbf{p}_1, D_1, \mathbf{q}_1), ..., (\mathbf{p}_j, D_j, \mathbf{q}_j)\}$.

Example 5. *In the MDP presented in Figure 1, the soft request set contains the single request:* $(\{p_0 = 0, p_1 = 0, p_2 = 1\}, 3, \{q_0 = 0, q_1 = 0, ...q_4 = 1\})$. *This request can be completed even if 3 time steps pass, but a cost will be incurred.*

Definition 2.7. Hard requests. The set of hard requests r_{hard} are the requests that must be completed before the deadline reaches 0. Formally, $r_{hard} = \{(\mathbf{p}_{j+1}, D_{j+1}, \mathbf{q}_{j+1}), (\mathbf{p}_{j+2}, D_{j+2}, \mathbf{q}_{j+2}), ..., (\mathbf{p}_W, D_W, \mathbf{q}_W)\}$.

Example 6. *In the MDP presented in Figure 1, the hard request set contains the single request:* $(\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}, 7, \{q_0 = 0, q_1 = 0, ...q_8 = 1\})$. *This request must be completed within 7 time steps.*

2.2 MDP Construction

Define the UAM scheduling MDP $M = (S, A, R, \delta)$ as follows.

Definition 2.8. State Space S. Define the state space as $S = \{r_{hard} \cup r_{soft}\} \cup \{s_{term}\}$ where there can exist only one request per route (without loss of generality), and s_{term} is the *terminal state* which is reached when a hard request misses its deadline. More formally, $S = \bigcup_{i=1}^{W} (\{p_0, p_1, ..., p_K\}, D_i, \{q_0, q_1, ..., q_M\}) \cup s_{term}$.

Example 7. *In the MDP presented in Figure 1, the state space is comprised of the two requests:* $(\{p_0 = 0, p_1 = 0, p_2 = 1\}, 3, \{q_0 = 0, q_1 = 0, ...q_4 = 1\})$ *and* $(\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}, 7, \{q_0 = 0, q_1 = 0, ...q_8 = 1\})$. *The computation time, deadline and inter-arrival times of these requests decrement until they are completed and/or replaced by a new instance of the request (due to the evolution of time) resulting in the observed state space. In addition to the different configurations of the requests, the state space also contains a terminal state (S13).*

Definition 2.9. Actions (A). Define an action $a_i \in A, i = 0, 1, 2, ..., W$, which corresponds to the scheduling of the i^{th} request r_i. Note that the 0^{th} action corresponds to idle time (e.g., no request is scheduled). Formally, the set of actions A is denoted $A = \{None, 1, 2, 3, ..., W\}$where W is the maximum number of requests (i.e., there can be at most one active request for each route at a time, where the total number of routes is W, as seen in definition 2.4).

Example 8. *In the MDP presented in Figure 1, the set of actions is* $A = \{None, 1, 2, ..., W\}$. *In state S1, a = 1 corresponds to working on* $(\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}, 7, \{q_0 = 0, q_1 = 0, ..., q_8 = 1\})$, *and a = 2 corresponds to working on* $(\{p_0 = 0, p_1 = 0, p_2 = 1\}, 3, \{q_0 = 0, q_1 = 0, ..., q_4 = 1\})$. *If* $a = None$, *none of the requests are worked on and the scheduler remains idle.*

Definition 2.10. Reward Function (R). Define the *reward function* R (where $R < 0$) as:

$$R(s, a, s') = \begin{cases} wJ_{soft}, & \text{if } \forall\ r_i = (p_0, ..., p_K, D_i, q_0, ..., q_M) \in s' | r_i \in r_{soft} \wedge D_i = 0 \wedge p_0 \neq 1 \\ J_{hard}, & \text{if } s' = s_{term} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where $J_{soft} < 0$ and $J_{hard} \ll J_{soft}$.

The first case corresponds to the deadline of the soft request r_i having elapsed ($D_i = 0$) and the request not having completed ($p_0 \neq 0$), which incurs a small (negative) penalty of J_{soft}. There can be up to w soft tasks whose deadlines elapse in a timestep, where $w \leq |r_{soft}|$. The second case corresponds to the deadline of the hard request r_i having elapsed ($D_i = 0$) and the request not having completed ($p_{0_i} \neq 0$), which incurs a very large (negative) penalty of J_{hard}. Otherwise, no reward is incurred ($R(s, a, s') = 0$).

Example 9. *As seen in Figure 1, a soft task misses its deadline when transitioning from S3 to state S4 or S5. In both cases, the deadline for the soft request reaches 0 while the request is incomplete (*$p_0 \neq 1$*). This results in a small negative reward of* $J_{soft} = -10$ *being accrued. A hard task misses its deadline in the transition from S12 to S13 in Figure 1. In this transition, S13 is the terminal state, resulting in a large negative reward of* $J_{hard} = -10000$ *being accrued.*

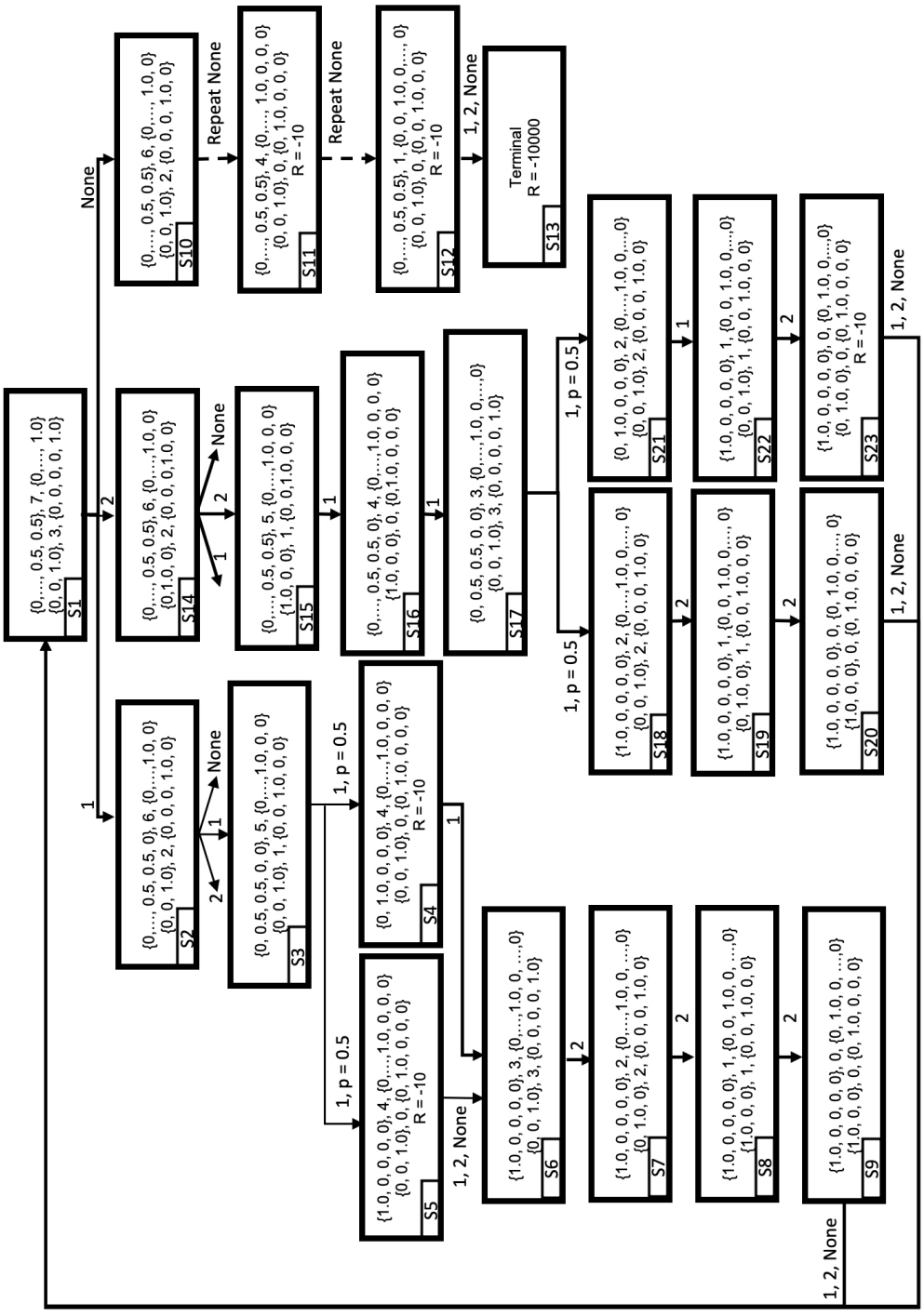


Figure 1: Markov Decision Process for Preemptive Scheduling Example

2.2.1 Transition Function

The construction of the transition function is presented as multiple individual cases that are complete when taken together. Formally, the *transition function* $\delta : S \times A \times S' \rightarrow [0,1]$ is defined as follows.

Case 1: Agent idle ($\mathbf{a} = None$) and no request is complete ($\mathbf{p_0} \neq \mathbf{1}$), has a probability p_1 of completing, or has reached its deadline ($\mathbf{D} \neq \mathbf{0}$). In this case, the deadline and inter-arrival time of all requests will decrement while the completion times remain the same. Formally, $\delta(s,a,s') = 1$ if:

$$\begin{aligned}
\forall\, r_i &= (\{p_0, p_1, ..., p_K\}, D_i, \{q_0, q_1, ..., q_M\}) \in s \mid p_1 = 0, D \neq 0, q_1 = 0, \\
a &= None, \\
r_i' &= (\{p_0', p_1', ..., p_K'\}, D_i', \{q_0', q_1', ..., q_M'\}) \mid \mathbf{p}_i' = \mathbf{p}_i, D_i' = D_i - 1, q_n' = q_{n+1}, q_M' = 0, n = 0, ..., M-1.
\end{aligned}$$

Example 10. *In the MDP in Figure 1, this case occurs when transitioning from S1 to S10. When the agent chooses the action None, the computation time remains the same for both requests (e.g., for request 1,* $\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\} \rightarrow \{p_0' = 0, p_1' = 0, p_2' = 0, p_3' = 0.5, p_4' = 0.5\}$*) while the deadline (e.g., for request 1,* $D_1 = 7 \rightarrow D_1' = 6$*) and inter-arrival time decrement (e.g., for request 1,* $\{q_0 = 0, q_1 = 0, ..., q_8 = 1\} \rightarrow \{q_0' = 0, q_1' = 0, ..., q_7' = 1, q_8' = 0\}$*).*

Case 2: Agent works on request ℓ and no request is complete, has a probability $p1$ of completing, or has reached its deadline. When the agent chooses to work on the ℓ^{th} request ($a = \ell \neq None$) and no request in the system is completed or has reached its deadline, the completion time, deadline, and inter-arrival time will decrement for the ℓ^{th} request, and all other requests will decrement their deadline and inter-arrival time while their completion time remains the same. Formally, $\delta(s,a,s')$ = 1 if:

$$\begin{aligned}
\forall r_i &= (\{p_0, p_1, ..., p_K\}, D_i, \{q_0, q_1, ..., q_M\}) \in s | p_0 = 0, D_i \neq 0, q_1 = 0, \\
a &= \ell, \\
r_\ell' &= (\{p_0', p_1', ..., p_K'\},\ D_\ell', \{q_0', q_1', ..., q_M'\}) \mid p_m' = p_{m+1}, D_\ell' = D_\ell - 1, q_n' = q_{n+1}\ , p_K' = 0, q_M' = 0 \\
&\quad \text{for } m = 0, ..., K-1 \text{ and } n = 0, ..., M-1, \\
r_i' &= ((\{p_0', p_1', ..., p_K'\}, D_i', \{q_0, q_1', ..., q_M'\}), i \neq \ell |\ \mathbf{p}_i' = \mathbf{p}_i, D_i' = D_i - 1, q_n' = q_{n+1}\ , q_M' = 0 \\
&\quad \text{for } n = 0, ..., M-1.
\end{aligned}$$

Example 11. *In the MDP in Figure 1, this case occurs when transitioning from S1 to S2. When the agent chooses action 1, the computation time decrements for the* 1^{st} *request (*$\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\} \rightarrow \{p_0' = 0, p_1' = 0, p_2' = 0.5, p_3' = 0.5, p_4' = 0\}$*) while the deadline (e.g., for request 1,* $D_1 = 7 \rightarrow D_1' = 6$*) and inter-arrival time (e.g., for request 1,* $\{q_0 = 0, q_1 = 0, ...q_8 = 1\} \rightarrow \{q_0' = 0, q_1' = 0, ..., q_7' = 1, q_8' = 0\}$*) decrement for both requests .*

When the completion time for a request r_ℓ in the system equals 1 with a non-zero probability ($p_{\ell_1} \neq 0$) and the agent chooses to work on that request ($A = \ell$), the request will be completed with a probability p_{ℓ_1}. On the other hand, the request will remain incomplete with probability $1 - p_{\ell_1}$. In this case, the non-zero probability entries of the next state will be increased by $p_{\ell_1}/|p_{nz} - 1|$, where $|p_{nz} - 1|$ is the remaining number of completion times with non-zero probability (i.e., if $p_{\ell_{m+1}} \neq 0$ then $p_{\ell_m}' = p_{\ell_{m+1}} + \frac{p_{\ell_1}}{|p_{nz}-1|}$, otherwise $p_{\ell_m}' = p_{\ell_{m+1}}$). The remaining requests all decrement their deadlines and inter-arrival times. Formally,$\delta(s,a,s') = 1 - p_{\ell_1}$ if:

$$
\begin{aligned}
\exists r_i &= (\{p_{i_0}, p_{i_1}, p_{i_2}, ..., p_{i_K}\}, D_i, \{q_{i_0}, q_{i_1}, ..., q_{i_M}\}) \in s | p_{i_1} \neq 0, D_i \neq 0, q_{i_1} = 0, \\
a &= \ell, \\
r'_\ell &= \left(\{p'_{\ell_0}, p'_{\ell_1}, p'_{\ell_2}, ..., p'_{\ell_K}\}, D_\ell, \{q'_{\ell_0}, q'_{\ell_1}, ..., q'_{\ell_M}\}\right) \mid \text{ if } p_{\ell_{m+1}} \neq 0 \text{ then } p'_{\ell_m} = p_{\ell_{m+1}} + \frac{p_{\ell_1}}{|p_{nz} - 1|}, \\
&\quad \text{otherwise } p'_{\ell_m} = p_{\ell_{m+1}}, D'_\ell = D_\ell - 1, q'_{\ell_n} = q_{\ell_{n+1}}, q'_M = 0 \text{ for } m = 0, ..., K-1 \text{ and} \\
&\quad n = 0, ..., M-1, \\
r'_i &= (\{p'_0, p'_1, ..., p'_K\}, D'_i, \{q'_0, q'_1, ..., q'_M\}), i \neq \ell |\ p'_i = p_i, D'_i = D_i - 1, q'_m = q_{m+1}, q'_M = 0\ , \\
&\quad m = 0, ..., M-1,
\end{aligned}
$$

and $\delta(s, a, s') = p_{\ell_1}$ if:

$$
\begin{aligned}
\exists r_i &= (\{p_{i_0}, p_{i_1}, p_{i_2}, ..., p_{i_K}\}, D_i, \{q_{i_0}, q_{i_1}, ..., q_{i_M}\}) \in s | p_{i_1} \neq 0, D_i \neq 0, q_{i_1} = 0, \\
a &= \ell, \\
r'_\ell &= (\{p'_{\ell_0}, p'_{\ell_1}, p'_{\ell_2}, ..., p'_{\ell_K}\}, D_\ell, \{q'_{\ell_0}, q'_{\ell_1}, ..., q'_{\ell_M}\}) \mid p'_{\ell_0} = 1, p'_{\ell_m} = 0, D'_\ell = D_\ell - 1, \\
&\quad q'_{\ell_n} = q_{\ell_{n+1}}, q'_M = 0 \text{ for } m = 1, ..., K \text{ and } n = 0, ..., M-1, \\
r'_i &= ((\{p'_0, p'_1, ..., p'_K\}, D'_i, \{q_0, q'_1, ..., q'_M\})\ , i \neq \ell |\ \mathbf{p}'_i = \mathbf{p}_i, D'_i = D_i - 1, q'_{i_n} = q_{i_{n+1}}, q'_M = 0, \\
&\quad \text{for } n = 0, ..., M-1.
\end{aligned}
$$

Example 12. *In the MDP presented in Figure 1, this case occurs when transitioning from S3 to S4 or S5. When the agent chooses action 1, the agent could transition to S5 with probability 0.5. In S5, the computation time becomes 0 for the* 1^{st} *request with probability 1 and all other probabilities in the distribution become 0, as the task has completed (e.g.,* $\{p_{1_0} = 0, p_{1_1} = 0.5, p_{1_2} = 0.5, p_{1_3} = 0, p_{1_4} = 0\} \rightarrow \{p'_{1_0} = 1, p'_{1_1} = 0, p'_{1_2} = 0, p'_{1_3} = 0, p_{1_4} = 0\}$*). On the other hand, the agent can also transition to S4 with probability 0.5. In S4, the computation time for the* 1^{st} *request becomes 1 with probability 1 (e.g,* $\{p_{1_0} = 0, p_{1_1} = 0.5, p_{1_2} = 0.5, p_{1_3} = 0, p_{1_4} = 0\} \rightarrow \{p'_{1_0} = 0, p'_{1_1} = 1, p'_{1_2} = 0, p'_{1_3} = 0, p'_{1_4} = 0\}$ *as* $p'_{l_1} = p_{l_2} + \frac{p_{l_1}}{|p_{nz}-1|} = 0.5 + \frac{0.5}{(2-1)} = 1$ *). In both states, the deadline (*$D_1 = 7 \rightarrow D'_1 = 6$*) and inter-arrival time decrement (e.g., for request 1,* $\{q_{1_0} = 0, q_{1_1} = 0, ...q_{1_6} = 1, q_{1_7} = 0, q_{1_8} = 0\} \rightarrow \{q'_{1_0} = 0, q'_{1_1} = 0, ..., q'_{1_5} = 1, q'_{1_6} = 0, q'_{1_7} = 0, q'_{1_8} = 0\}$*).*

Case 4: Agent chooses to schedule a completed request ($\mathbf{a} = \ell \wedge \mathbf{p_0} = \mathbf{1} \wedge \mathbf{q_1} \neq \mathbf{1}$)). When the completion time for a request r_ℓ in the system equals 0 with a probability of 1 ($p_0 = 1$) and the agent chooses to work on the request ($A = \ell$), the action $A = \ell$ will be treated like the *None* action (i.e., **Case 1**). This causes the inter-arrival times of requests and all non-zero deadlines to decrement. The completion time of all requests remains the same. Deadlines of soft requests which are zero remain zero, and if a deadline of a hard request is zero, see **Case 6**. Formally, $\delta(s, a, s') = 1$ if:

$$
\begin{aligned}
\exists r_\ell &= (\{p_0, p_1, ..., p_K\}, D_\ell, \{q_0, q_1, ..., q_M\}) \in s | p_0 = 1, q_1 = 0, \\
a &= \ell, \\
\forall r'_i &= (\{p'_0, p'_1, ..., p'_K\}, D'_i, \{q'_0, q'_1, ..., q'_M\}) \in s',\ \mathbf{p}'_i = \mathbf{p}_i, \text{ if } D_i \neq 0,\ D'_i = D_i - 1, \\
&\quad \text{otherwise } (D'_i = D_i = 0 \wedge r_i \in r_{soft}), q'_n = q_{n+1}\ , q'_M = 0 \text{ for } n = 0, ..., M-1.
\end{aligned}
$$

Case 5: Soft request(s) missed their deadline(s), and are not yet replaced ($\mathbf{r_i} \in \mathbf{r_{soft}} \wedge \mathbf{p_0} \neq \mathbf{1} \wedge \mathbf{D_i} = \mathbf{0} \wedge \mathbf{q_1} = \mathbf{0}$). In this case, the soft requests that missed their deadlines have a deadline of 0 in their

next state. The deadlines of all other requests and the inter-arrival times of every request will decrement. The completion time of the request selected (e.g., $a = \ell$) will also decrement. All non-selected requests will keep their completion times the same. Formally, $\delta(s,a,s') = 1$ if:

$$\begin{aligned}
\exists r_i &= (\{p_0, p_1, ..., p_K\}, D_i, \{q_0, q_1, ..., q_M\}) \in s \mid D_i = 0, q_1 = 0 \\
a &= \ell, \\
r'_\ell &= (\{p'_0, p'_1, ..., p'_K\}, D'_\ell, \{q'_0, q'_1, ..., q'_M\}) \in s' \mid p'_m = p_{m+1}, p'_K = 0, \\
&\quad \text{if } (D_\ell = 0 \wedge r_\ell \in r_{soft}) \vee (r_\ell \in r_{hard} \wedge p_0 = 1) \\
&\quad \text{then } D'_\ell = 0, \text{ otherwise } D'_\ell = D_\ell - 1, q'_n = q_{n+1}, q'_M = 0 \text{ for } m = 0, ..., K-1, \\
&\quad \text{and } n = 0, ..., M-1, \\
r'_i &= (\{p'_0, p'_1, ..., p'_K\}, D'_i, \{q_0, q'_1, ..., q'_M\}), i \neq \ell \mid \mathbf{p}'_m = \mathbf{p}_m, \\
&\quad \text{if } (D_i = 0 \wedge r_i \in r_{soft}) \vee (r_i \in r_{hard} \wedge p_0 = 1) \text{ then } D'_i = 0, \\
&\quad \text{otherwise } D'_i = D_i - 1, q'_n = q_{n+1}, q'_M = 0 \text{ for } n = 0, ..., M-1, \text{ and} \\
R &= -10, \forall r_i \in r_{soft} \text{ with } D'_i = 0 \wedge D_i \neq 0.
\end{aligned}$$

Example 13. *Consider S23 in the MDP in Figure 1 where the hard task* $r_1 = (\{p_0 = 1.0, p_1 = 0, p_2 = 0, p_3 = 0, p_4 = 0\}, 0, \{q_0 = 0, q_1 = 1, q_2 = 0, q_3 = 0, ..., q_9 = 0\})$ *is complete and the soft task* $r_2 = (\{p_0 = 0, p_1 = 1, p_2 = 0\}, 0, \{q_0 = 0, q_1 = 1, q_2 = 0, q_3 = 0, q_4 = 0\})$ *has missed its deadline and incurred a cost of* $R = -10$*. Unlike in Figure 1, if the soft request was not scheduled for replacement (e.g., if* $q_1 \neq 1$*) and the soft request is worked on, the next state would complete the task* $r'_2 = (\{p'_0 = 1.0, p'_1 = 0, p'_2 = 0\}, 0, \{q'_n = q_{n+1}, ...\})$*. Note if the completed hard task was not scheduled for replacement, it would decrement normally (e.g.,* $r'_1 = (\{p'_0 = 1.0, p'_1 = 0, p'_2 = 0, p'_3 = 0, p'_4 = 0\}, 0, \{q'_n = q_{n+1}, ...\})$*).*

Case 6: Hard request misses deadline ($\mathbf{D_i = 0 \wedge r_i \in r_{hard} \wedge p_1 \neq 0}$). When a request has a hard deadline D and the trip is incomplete by D, the next state is the terminal state with $p = 1$. Formally, $\delta(s,a,s') = 1$ if:

$$\begin{aligned}
\exists\, r_i &= (\{p_0, p_1, ..., p_K\}, D_i, \{q_0, q_1, ..., q_M\}) \in s | p_0 \neq 1, D_i = 0, r_i \in r_{hard}, \\
\forall\, a &\in A, \\
s' &= s_{term}.
\end{aligned}$$

Example 14. *In the MDP in Figure 1, this case occurs when transitioning from S12 to S13. From S12, the deadline for* r_1 *will reach* 0 *regardless of the action taken. When the agent chooses any action in A, it transitions to the terminal state S13 with probability 1.0.*

Case 7: New instance of request $\mathbf{r_\ell}$ arrives with probability $\mathbf{q_{\ell_1}}$. When the inter-arrival time for a request r_ℓ in the system equals 1 with a non-zero probability q_{ℓ_1}, the request will be replaced with a new instance with a probability q_{ℓ_1} in the next timestep. On the other hand, the request will persist with probability $(1 - q_{\ell_1})$. In this case, the remaining non-zero probability entries of the next state will be increased by $q_{\ell_1}/|q_{nz} - 1|$ where $|q_{nz}|$ is the number of inter-arrival times with non-zero probability. Note that multiple requests may have new instances arriving simultaneously. Formally, $\delta(s,a,s') = 1 - q_{\ell_1}$ if:

$$
\begin{aligned}
\exists r_\ell &= (\{p_{\ell_0}, p_{\ell_1}, p_{\ell_2}, ..., p_{\ell_K}\}, D_\ell, \{q_{\ell_0}, q_{\ell_1}, ..., q_{\ell_M}\}) \in s | q_{\ell_1} \neq 0, \\
a &= j, \\
r'_\ell &= (\{p'_{\ell_0}, p'_{\ell_1}, p'_{\ell_2}, ..., p'_{\ell_K}\}, D'_\ell, \{q'_{\ell_0}, q'_{\ell_1}, ..., q'_{\ell_M}\}) \mid \text{ if } (\ell = j) \wedge (p_0 \neq 1), p'_{\ell_m} = p_{\ell_{m+1}}, p_{\ell_K} = 0, \\
&\quad \text{otherwise } \mathbf{p}'_\ell = \mathbf{p}_\ell, \text{ if } D_\ell \neq 0,\ D'_\ell = D_\ell - 1, \text{ otherwise } D'_\ell = 0, \text{ if } q_{\ell_{n+1}} \neq 0 \\
&\quad \text{then } q'_{\ell_n} = q_{\ell_{n+1}} + \frac{q_{\ell_1}}{|q_{nz} - 1|}, \\
&\quad \text{otherwise } q'_{\ell_n} = q_{\ell_{n+1}}, q'_{\ell_M} = 0, \text{ for } m = 0, ..., K-1, \text{ and } n = 1, ..., M-1.
\end{aligned}
$$

Example 15. *In the MDP presented in Figure 1, this case can be seen when transitioning from S4 or S5 to S6. When the agent chooses any action, the agent transitions to S6. In S6, the inter-arrival time becomes 0 for the* 1^{st} *request. This results in the* 1^{st} *request being replaced by a new request in its initial configuration* $(\{p_0 = 0, p_1 = 0, p_2 = 1\}, 0, \{q_0 = 0, q_1 = 1, ...q_4 = 0\}) \rightarrow (\{p_0 = 0, p_1 = 0, p_2 = 1\}, 3, \{q_0 = 0, q_1 = 0, ...q_4 = 1\})$.

Example 16. *An example of two new requests arriving simultaneously occurs in states S9, S20, and S23. Under any action chosen in those three states, a hard request and a soft request will arrive. The initial configuration of S1 is then recovered* $[(\{p_0 = 0, p_1 = 0, p_2 = 1\}, 0, \{q_0 = 0, q_1 = 1, ...q_4 = 0\}), (\{p_0 = 1, p_1 = 0, p_2 = 0, p_3 = 0, p_4 = 0\}, 0, \{q_0 = 0, q_1 = 1, ...q_8 = 0\})] \rightarrow [(\{p_0 = 0, p_1 = 0, p_2 = 1\}, 3, \{q_0 = 0, q_1 = 0, ...q_4 = 1\}), (\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}, 7, \{q_0 = 0, q_1 = 0, ...q_8 = 1\})]$. *This is shown in the loops from S9, S20, and S23 to S1.*

Case 8: Agent in terminal state (s = s_term). In this case, the next state will also be the terminal state regardless of action taken. Formally, $\delta(s, a, s') = 1$ if:

$$
\begin{aligned}
s &= s_{term} \\
\forall\, a &\in A \\
s' &= s_{term}
\end{aligned}
$$

Example 17. *In the MDP in Figure 1, this case occurs when transitioning from S13. When the agent chooses any action from S13, the agent transitions back to S13 with probability 1.0.*

2.3 Preemptible vs. Non-Preemptible Scheduling

Two types of MDPs are defined. A *preemptible* MDP is an MDP without any constraints on actions it can take at any given state. A request can be stopped before it is completed and another request can be started without any consequences. Unlike a preemptible MDP, a *non-preemptible* MDP may have a limited set of actions it can take at any given state. Specifically, an agent traversing a non-preemptible MDP must finish a request before starting a new request, and no idle time can be inserted into the completion of the request (e.g., $a \neq None$). If a new instance of a request arrives before an agent completes the current request, the agent will have the option to start a different request instead of working on the new instance.

2.3.1 Non-Preemptible MDP State Space

The state space for non-preemptible MDPs can be partitioned into two subsets: (1) Restricted states (S_r), in which an agent only has access to one action and (2) Unrestricted states (S_u), in which an agent has

access to more than one action. The implementation of the non-preemptible MDP abstracts restricted states from the preemptible MDP, resulting in a smaller state-space (e.g., successor states resulting from the repeated application of a single action are abstracted into a single state).

Example 18. *In the non-preemptible MDP based on the MDP presented in Figure 1, S1 would be considered an unrestricted state since the traversing agent is free to choose any action $a \in A$. However, once the agent transitions to S2 by choosing action 1, it is unable to choose any request but 1 until r_1 is complete. As a result, S2 would be considered a restricted state. In this implementation S2, S3, and S4 would be abstracted as restricted states and the agent would transition directly from S1 to S5 or S6 after choosing action 1.*

3 Generating the Safe Scheduler

The high-level process for solving a UAM scheduling problem is shown in Figure 2. The process begins by constructing an MDP given a set of routes and requests as described in Subsection 2.2 (Construction). The MDP is then pruned to remove terminal cases so agents can safely traverse the MDP while sampling (Pruning). The agent traverses the pruned MDP to sample the completion and inter-arrival time distributions (Sampling). After sampling the distributions, the agent can create an estimate MDP (Learning), which it uses to find the optimal policy by employing value iteration or MCTS (Solving).

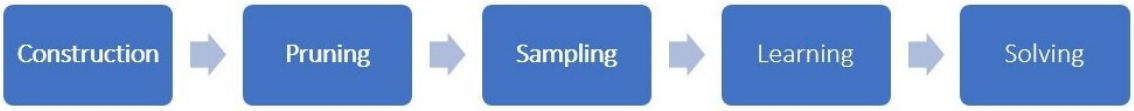


Figure 2: Process to solve the UAM Scheduling Problem

3.1 Pruning the MDP

The scheduler begins by pruning the MDP to remove any terminal states from the system. A backwards reachability algorithm is used to prune states and actions that have non-zero probability of leading to the terminal state. If all actions for a given state are pruned, then the state is also pruned from the MDP, and that state is labelled as a terminal state. The same pruning step is then performed for all actions and states which lead to the newly labelled terminal state. For example, in the MDP figure diagram, the agent will begin by pruning the Terminal state ($S13$). From the terminal state, the algorithm will backtrace to state $S12$ and prune the actions $\{1, 2, None\}$, as they all lead to the Terminal state. Since all the actions at $S12$ are pruned, the algorithm will prune $S12$ from the system and repeat the process for all actions and states which lead to $S12$. As a result, the algorithm will prune any intermediate states between $S11$ and $S12$ as they inevitably lead to the Terminal state. At $S11$, the pruning step will remove the None and 2 action as choosing any action other than 1 will lead to a pruned state and, by extension, the Terminal State. This process applies to both a preemptible and non-preemptible MDP.

3.2 Sampling the MDP

After pruning, the scheduling agent begins traversing the pruned MDP. When sampling a specific request, the agent will choose to work on the specific request at all time steps except when prevented by the pruning step. When the request is completed, the agent increments a counter corresponding to the number

of time steps required to finish the request. The agent repeats this process for a given number of samples. Once the specified sample number is reached, the agent normalizes the counters across the number of samples to obtain an estimated probability distribution. The agent performs the same process when estimating the inter-arrival time distribution and records when a new instance of the same request appears. The agent repeats the sampling process for all requests in the system to have an accurate estimate of the completion and inter-arrival times of all requests. The number of samples is determined by the formula outlined in [10]. Once all requests have been sampled, the agent uses the estimated distributions to create an estimate request system and an estimate MDP. By running value iteration or MCTS on the estimate MDP, the scheduling agent can choose the near-optimal action at each time step.

The difference in probability distributions between the pruned and unpruned MDPs comes from paths that would lead to the terminal state, which should not be taken. By removing those paths, the estimated MDP only captures the safe traces of the actual MDP, and thus the MDP is safely learned.

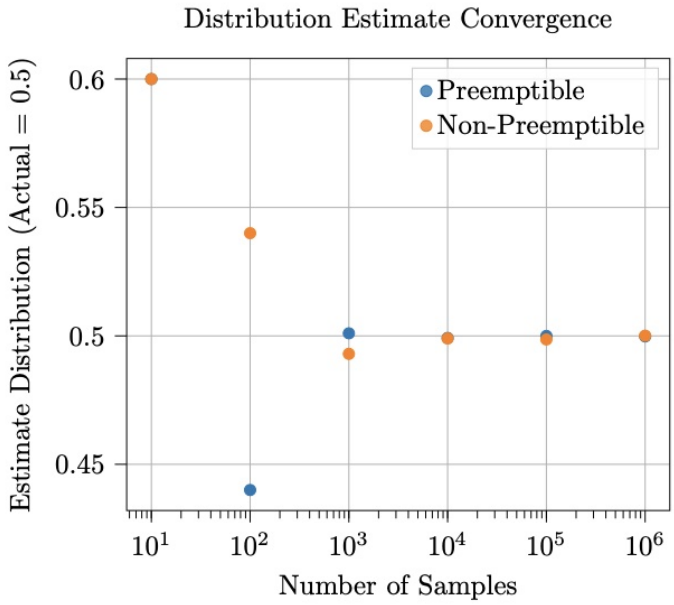


Figure 3: Delay Distribution for Priority Route with Completion Time={1,2}

The accuracy of the sampling algorithm is shown in Figure 3. A preset MDP was used as a plant for experimentation to gauge how well the agent could estimate the MDP. In [4], the authors derive that $y = r * \left\lceil \frac{1}{2\varepsilon^2}(\ln 2r - \ln \gamma) \right\rceil$ in order to establish the relationship between ε, the percentage to which the estimated MDP converges to the actual MDP, and y, the numbers of samples taken over the plant. In this expression, r represents the size of the support of the estimated distribution and $1-\gamma$ is the probability of the estimated model being within ε of the actual model. A value of 1000 samples is taken in this work, with $1-\gamma = 0.9$ and $r = 2$, which yields epsilon as approximately 0.0607. The estimated MDP is within 6.07 % of the actual MDP and has approximately 94% accuracy. The completion time distribution of $\{q_0 = 0, q_1 = 0.5, q_2 = 0.5\}$ was the configuration being estimated and a set number of samples (1000) was employed. As the number of samples increased, the scheduler converged on the correct distribution in the preemptible and non-preemptible implementations. The MDP can be estimated using these approximations to the inter-arrival and completion time distributions.

3.3 Learning the MDP

After sampling the MDP, estimate probability distributions for computation time and inter-arrival time are created. The calculation for the estimate distributions is: # samples for a given time/total samples. For example, if a sampling agent took 10 samples from a request in a system and recorded that 6 of those samples took 3 time steps to complete, it would estimate the completion time of 3 of the request as: $p_3 \in T_{est} = 6/10 = 0.6$. After creating estimates for the computation and inter-arrival times for all requests, the agent creates the estimate request set: $\{\hat{r}_1, \hat{r}_2, ..., \hat{r}_n\} = \{(\hat{\mathbf{p}}_1, D_1, \hat{\mathbf{q}}_1), (\hat{\mathbf{p}}_2, D_2, \hat{\mathbf{q}}_2), ..., (\hat{\mathbf{p}}_n, D_n, \hat{\mathbf{q}}_n)\}$. After estimating the request set, the construction process explained in Section 2.2 is used to create an estimated MDP. Since the agent cannot miss hard deadlines during sampling, it cannot gain a comprehensive estimate of the reward function. Thus, the reward function values are provided prior to sampling.

4 Solving for the Optimal Policy of the Learned MDP

After estimating the MDP model, the next step is to find the optimal reward while traversing the MDP. A policy $\pi(s) \rightarrow a$ is defined as mapping a state s to an action a, and the goal is to find the optimal policy that maximizes (or minimizes) an objective function over the MDP's time horizon, τ. An optimal policy π for an infinite time horizon MDP can maximize the expected discounted total reward as follows:

$$\max V_\pi(s) = \max \left\{ \lim_{\tau \to \infty} E_{\pi,s} \left[\sum_{t=0}^{\tau} \gamma_t R(s'_t | s_t, \pi(a_t)) \right] \right\} \tag{2}$$

where γ_t is the discount factor at time t, s_t and a_t are the state and action taken at decision time t, and $E[\cdot]$ is the expectation function. This is a stationary policy for the infinite-horizon discounted reward MDP due to the stationary nature of the transition probabilities and (bounded) rewards. The maximal achievable reward $v^*(s)$ at every state s is found recursively by solving Bellman's equation:

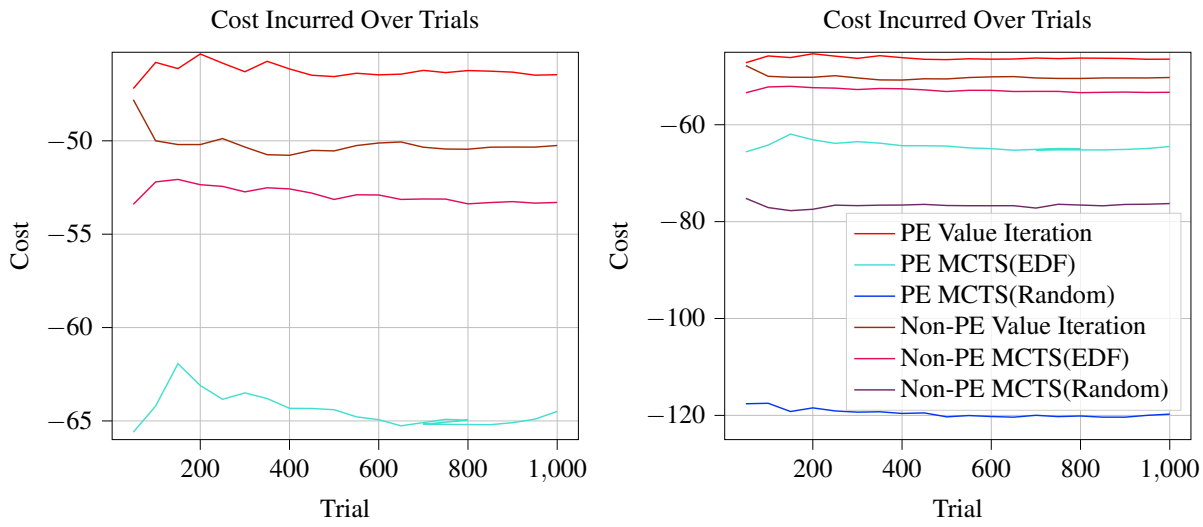
$$v_t^*(s) = \max_{a \in A} \left[R_t(s,a) + \sum_{s' \in S} P_t(s'_t | s_t, a) v_{t+1}^*(s') \right] \tag{3}$$

This can be done using several means: (1) Value Iteration (VI), which is optimal but computationally intensive and (2) Monte Carlo Tree Search (MCTS), which approaches optimality for large sample sizes. Standard treatments and implementations for solving Bellman's equation using VI can be found in [21], [11]. The MCTS approach is explained below.

4.1 MCTS

Unlike VI, which finds the optimal action for every state within an MDP, MCTS returns the optimal action for a given input state. The algorithm begins at the input state and deterministically takes one of the available actions. The functional steps used by the MCTS algorithm are outlined in [4]. Since the MCTS algorithm's primary input parameter is the search depth and number of samples, its runtime is independent of state space size. Thus, the scheduling agent can choose actions with a consistent run time across request configurations.

The UAM MCTS algorithm uses Earliest-Deadline First (EDF) as the MCTS simulation policy. The EDF policy deterministically returns an action at each state in the MDP. When possible, the EDF policy chooses the hard request with the shortest deadline. If all hard requests have been completed, the policy chooses the soft request with the shortest deadline. This policy has been shown to be optimal when



(a) Comparison of value iteration optimal policy and MCTS approximation (zoom of Fig 4b y-axis)

(b) Comparison of EDF request selection strategy to random selection in MCTS approximation

Figure 4: Average Cost for Preemptible and Non-preemptible UAM Scheduling for Fig. 1 Request System

scheduling processor utilization for all tasks is less than 1 [2],[6]. The EDF policy is only optimal when the tasks are preemptible and a feasible schedule exists, i.e., a schedule that does not violate the deadline of a hard request [6]. By using EDF as the MCTS simulation policy, the scheduling agent can obtain a conditionally-optimal baseline for reward estimation at a given state and action.

4.2 Solving the MDP: VI vs MCTS

VI and MCTS were used to approximate the optimal policy over the learned MDP. The route configuration seen in Fig. 1 for both the preemptible and non-preemptible scheduling MDP implementation were evaluated. The scheduling agent ran through ten traversals of each MDP while choosing optimal actions. To account for probabilistic variation, 1000 trials were conducted, with each trial recording the total reward across 10 traversals, with an average being taken every 50 trials.

The optimal preemptible policy found through VI, as shown in Fig. 4a, converges to a cost of approximately -46 (averaged over 10 MDP traversals and 1000 trials). The non-preemptible optimal policy obtained through VI, also shown in Fig. 4a, converges to a cost of about -50 (averaged over 10 MDP traversals and 1000 trials). Since the non-preemptible scheduler is less flexible (i.e., has fewer states) than the preemptible, its optimal policy accrues a larger cost over time. The MCTS approximation to the optimal policy for both the preemptible and non-preemptible MDP (see Figures 4a,4b) is strictly inferior to the one obtained by VI, but it forms a near approximation. Increasing the depth and number of MCTS samples has that policy approach the optimal policy of VI, which is observed in the convergence of the MCTS average cost to that of VI.

4.3 Near-Optimality of Average Cost: MCTS Selection Strategy

The strategy used to select the next action (i.e., request) to work on strongly affects the ability of the MCTS algorithm to approximate the optimal policy obtained through VI. The use of random action

selection for MCTS, shown in Fig. 4b, exemplifies how the EDF selection strategy, coupled with preventing the agent from working on completed requests in the preemptible case, allows MCTS to choose the optimal action more frequently. The average cost of random MCTS is significantly larger than the average cost of EDF MCTS schedulers. This is expected, as the EDF selection strategy results in more accurate cost estimates than random MCTS at each timestep, resulting in optimal actions being selected more frequently.

State space abstraction allows the MCTS non-preemptible near-optimal policy to outperform the MCTS preemptible near-optimal policy for both EDF and Random (see Fig. 4b). Since the simulation depth parameter of MCTS is kept constant, the MCTS algorithm is able to step farther into the future for the non-preemptible MDP thereby enabling optimal actions to be chosen more frequently at each time step. The preemptible scheduler has a large branching factor and several intermediate states. This makes it difficult to obtain an accurate future cost estimate using a random policy in the preemptible case.

5 Results and Discussion

The scalability of the approach for both preemptible and non-preemptible MDPs is evaluated. The optimal policy is obtained through VI, run with a convergence parameter of 0.99, over a variety of task configurations as well as demand and delay distributions. This is done on a Dell XPS 13 Windows Processor with an 11^{th} generation Intel Single Core i7 at 3.0 GHz. The baseline request configuration is defined as two deterministic tasks: $r_1 \in r_{hard} = (\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 1.0\}, 7, \{q_0 = 0, q_1 = 0, ..., q_8 = 1\})$, $r_2 \in r_{soft} = (\{p_0 = 0, p_1 = 0, p_2 = 1\}, 3, \{q_0 = 0, q_1 = 0, ..., q_4 = 1\})$.

5.1 Scalability over Number of Routes

# Routes	Preemptible Average Time (s)	Preemptible # States	Non-Preemptible Average Time (s)	Non-Preemptible # States
1 Hard & 1 Soft	1.438	47	0.212	18
1 Hard & 2 Soft	1033.546	82	126.102	23
1 Hard & 3 Soft	5090.536	131	375.913	28

Table 1: Scalability with Respect to Number of Routes for Value Iteration

The scalability of the technique with respect to increasing the number of requests is explored. The baseline configuration is modified by adding additional copies of the soft request to the system, as inserting unmodified copies of the hard request can lead to unschedulable systems. The run time of VI (in seconds), the number of unrestricted states in the MDP, and the number of iterations the VI algorithm took to converge are then recorded. Table 1 shows a linear-like increase in the state size as the number of requests increases. Due to the abstracted restricted states, the state space of the non-preemptible MDP grows more slowly than the preemptible MDP. Since the time complexity of value iteration is $O(|S^2A|)$, the non-preemptible MDP has a shorter run time than the preemptible MDP.

In the 1H & 1S case, the MDP can be traversed without missing any deadlines. This allows the VI algorithm to converge in a few iterations since the optimal expected future reward is 0. Optimally traversing the other configurations results in a guaranteed negative reward, which leads to a larger number of iterations. The increase in the number of states and iterations leads to a larger computation time as the number of requests increased, making MCTS a necessary alternative once VI becomes intractable.

5.2 Uncertain Trip Delay (Completion) and Demand (Inter-Arrival) Time Distributions

Completion Time (Delay) Values	*P*(Delay)	Preemptible Average Time (s)	Preemptible # States	Non-Preemptible Average Time (s)	Non-Preemptible # States
$\{3\}$	$p_3 = 1$	1.438	47	0.212	18
$\{3,4\}$	$p_{3-4} = 0.5$	200.694	54	29.636	18
$\{1,2,3,4\}$	$p_{1-4} = 0.25$	225.12	59	37.715	21

Table 2: Increasing Uncertainty over Delay Distribution (Completion Time) for Hard Requests

Scalability with respect to the completion time distribution support (e.g., delay) is also investigated. The support size of the trip completion time (delay) probability distribution of the hard request is varied. Three different support sizes over completion time are used: (1) $\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 1.0\}$, (2) $\{p_0 = 0, p_1 = 0, p_2 = 0, p_3 = 0.5, p_4 = 0.5\}$, and (3) $\{p_0 = 0, p_1 = 0.25, p_2 = 0.25, p_3 = 0.25, p_4 : 0.25\}$. Table 2 shows a marked increase in MDP state space size. The increased number of possible completion times leads to a larger number of potential negative reward conditions, resulting in an increased number of iterations for VI to converge.

Increasing the range of the completion time support function to include 4 values introduces the case where the agent will continuously miss the soft request deadline with a probability of p_4. The introduction of this persistent negative reward leads to a larger number of iterations for convergence and results in a larger run time when combined with the larger state space.

Inter-Arrival Time (Demand) Values	*P*(Demand)	Preemptible Average Time (s)	Preemptible # States	Non-Preemptible Average Time (s)	Non-Preemptible # States
$\{8\}$	P = 1	1.438	47	0.212	18
$\{8,9\}$	P = 0.5	101.663	201	27.635	75
$\{8,9,10,11\}$	P = 0.25	130.924	219	47.466	87

Table 3: Increasing Uncertainty over Demand Distribution (Inter-Arrival Time) for Hard Requests

Similarly, scalability with respect to the support size of the inter-arrival time (demand) distribution is also studied. Three different inter-arrival support sizes are examined: (1) $\{q_0 = 0, q_1 = 0, ..., q_8 = 1.0\}$, (2) $\{q_0 = 0, q_1 = 0, ..., q_8 = 0.5, q_9 = 0.5\}$, and (3) $\{q_0 = 0, q_1 = 0, ..., q_8 = 0.25, q_9 = 0.25, q_10 = 0.25, q_11 : 0.25\}$. Table 3 shows that MDP state space increases in a polynomial fashion across the inter-arrival time probability distribution support function. Changing the inter-arrival time distribution impacts when new request instances arrive, and the introduction of new requests at different time steps leads to

longer branches in the MDP that encompass the entire completion time and inter-arrival time of the new request. The completion time distribution only impacts the completion time of the requests and thus causes a smaller increase in the state space. However, there exists a way to traverse both the preemptible and non-preemptible MDPs without missing any deadlines in all three cases. This allows VI to converge in a small number of iterations resulting in smaller run times when compared to Table 3.

6 Conclusion

A safe, non-preemptible scheduler for a UAM route planning system was generated for both hard and soft requests. The UAM trip request scheduling system was modelled as an MDP, and safe learning was used to estimate the demand and delay distributions for the requests. The synthesized estimate MDP was then used to identify the near-optimal policy that minimized the average cost incurred by missed soft deadlines while guaranteeing no hard deadlines were missed using MCTS (EDF). The approach was shown to be scalable in terms of number of routes and size of both delay and demand distributions. A comparison of the results with Value Iteration and MCTS (Random) was also performed. Future work will center on improving the scalability of the approach with respect to increasing numbers of hard requests and applying the approach to generate schedulers from existing UAM schedule databases.

7 Downloads

The code to run the example can be found at: https://github.com/suryakmurthy/UAM_scheduler.

References

[1] H. Aydin, R. Melhem, D. Mosse & P. Mejfa-Alvarez (1999): *Optimal reward-based scheduling of periodic real-time tasks*. In: *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054)*, pp. 79–89, doi:10.1109/REAL.1999.818830.

[2] Sanjoy K. Baruah, Rodney R. Howell & Louis Rosier (1990): *Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic, Real-Time Tasks on One Processor*. *Real-Time Systems* 2, pp. 301–324, doi:10.1007/BF01995675.

[3] Alan Burns (1991): *Scheduling hard real-time systems: a review*. *Software Engineering Journal* 6(3), pp. 116–128, doi:10.1049/sej.1991.0015.

[4] Damien Busatto-Gaston, Debraj Chakraborty, Shibashis Guha, Guillermo A. Pérez & Jean-François Raskin (2021): *Safe Learning for Near-Optimal Scheduling*. In: *Quantitative Evaluation of Systems: 18th International Conference, QEST 2021, Paris, France, August 23–27, 2021, Proceedings*, Springer-Verlag, Berlin, Heidelberg, p. 235–254, doi:10.1007/978-3-030-85172-9_13.

[5] Giorgio C Buttazzo, Marko Bertogna & Gang Yao (2012): *Limited preemptive scheduling for real-time systems. a survey*. *IEEE transactions on Industrial Informatics* 9(1), pp. 3–15, doi:10.1109/TII.2012.2188805.

[6] H. Chetto & M. Chetto (1989): *Some Results of the Earliest Deadline Scheduling Algorithm*. *IEEE Transactions on Software Engineering* 15(10), pp. 1261–1269, doi:10.1109/TSE.1989.559777.

[7] Robert Davis & Alan Burns (2009): *A survey of hard real-time scheduling algorithms and schedulability analysis techniques for multiprocessor systems*. Technical Report, University of York, Department of Computer Science, YCS-2009-443.

[8] Robert Davis & Alan Burns (2011): *A survey of hard real-time scheduling for multiprocessor systems*. *ACM computing surveys (CSUR)* 43(4), pp. 1–44, doi:10.1145/1978802.1978814.

[9] Sudarshan K Dhall & Chung Laung Liu (1978): *On a real-time scheduling problem*. Operations research 26(1), pp. 127–140, doi:10.1287/opre.26.1.127.

[10] Gilles Geeraerts, Shibashis Guha & Jean-François Raskin (2018): *Safe and Optimal Scheduling for Hard and Soft Tasks*. In: 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018), Leibniz International Proceedings in Informatics (LIPIcs) 122, pp. 36:1–36:22, doi:10.4230/LIPIcs.FSTTCS.2018.36.

[11] Abhijit Gosavi (1997): *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Operations Research/ Computer Science Interfaces Series 25, doi:10.1007/978-1-4757-3766-0.

[12] K.S. Hong & J.Y.-T. Leung (1992): *On-line scheduling of real-time tasks*. IEEE Transactions on Computers 41(10), pp. 1326–1331, doi:10.1109/12.166609.

[13] Sang Hyun Kim (2019): *Receding horizon scheduling of on-demand urban air mobility with heterogeneous fleet*. IEEE Transactions on Aerospace and Electronic Systems 56(4), pp. 2751–2761, doi:10.1109/TAES.2019.2953417.

[14] Imke C Kleinbekman, Mihaela A Mitici & Peng Wei (2018): *eVTOL arrival sequencing and scheduling for on-demand urban air mobility*. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), IEEE, pp. 1–7, doi:10.1109/DASC.2018.8569645.

[15] John P. Lehoczky, Lui Sha & Jay K. Strosnider (1987): *Enhanced Aperiodic Responsiveness In Hard Real-Time Environments.* In: RTSS 1987, IEEE, pp. 261–270.

[16] Joseph Y.T. Leung & M. L. Merrill (1980): *A note on preemptive scheduling of periodic, real-time tasks*. Information Processing Letters 11(3), pp. 115–118, doi:10.1016/0020-0190(80)90123-4.

[17] Kwei-Jay Lin, Swaminathan Natarajan & Jane W-S Liu (1987): *Imprecise results: Utilizing partial computations in real-time systems*. Technical Report, NASA.

[18] C. L. Liu & James W. Layland (1973): *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*. J. ACM 20(1), p. 46–61, doi:10.1145/321738.321743.

[19] Steve Paul & Souma Chowdhury (2022): *A Graph-based Reinforcement Learning Framework for Urban Air Mobility Fleet Scheduling*. In: AIAA AVIATION 2022 Forum, p. 3911, doi:10.2514/6.2022-3911.vid.

[20] Priyank Pradeep & Peng Wei (2018): *Heuristic approach for arrival sequencing and scheduling for eVTOL aircraft in on-demand urban air mobility*. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), IEEE, pp. 1–7, doi:10.1109/DASC.2018.8569225.

[21] Martin L. Puterman (1994): *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st edition. John Wiley & Sons, Inc., USA, doi:10.1002/9780470316887.

[22] Syed Arbab Mohd Shihab, Peng Wei, Daniela Sofia Jurado Ramirez, Rodrigo Mesa-Arango & Christina Bloebaum (2019): *By schedule or on demand?-a hybrid operation concept for urban air mobility*. In: AIAA Aviation 2019 Forum, p. 3522, doi:10.2514/6.2019-3522.

[23] Marco Spuri & Giorgio C. Buttazzo (2004): *Scheduling aperiodic tasks in dynamic priority systems*. Real-Time Systems 10, pp. 179–210, doi:10.1007/BF00360340.