Scheduling for Urban Air Mobility using Safe Learning

Natasha A. Neogi
NASA Langley Research Center

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

Suda Bharadwaj
Skygrid Inc.

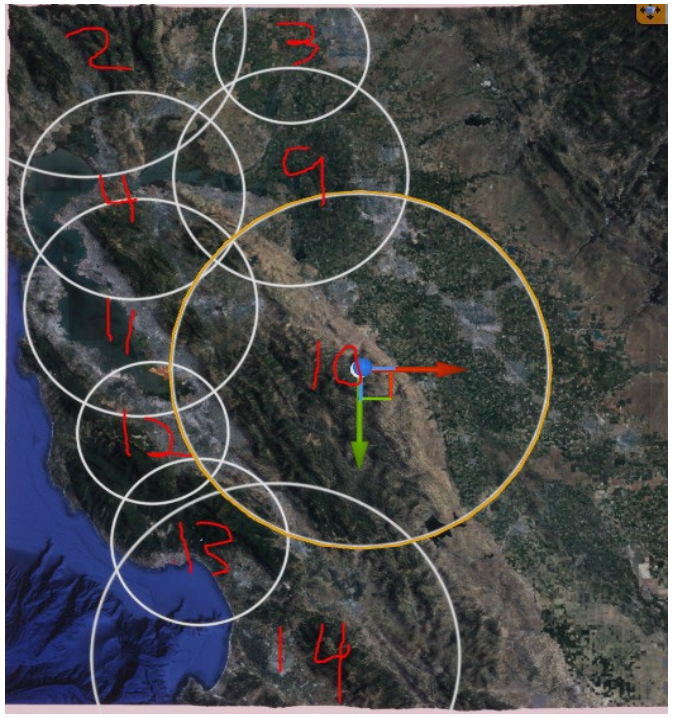
Agenda

1. **UAM Background and Research Question**
2. **Research Approach and Literature Review**
3. **Developing MDP model of the UAM System**
4. **Solving the MDP - Constructing, Sampling and Solving the MDP**
5. **Conclusions and Future Work**

1

Urban Air Mobility (UAM) Background

- Goal of Urban Air Mobility is to improve public transit
- Current System Setup:
  - Vertiports manage takeoff/landing
  - Vertihubs are composed of vertiports and manage the airspace
  - Towers provide travel requests
  - Vehicles complete the requests



https://earth.google.com/web/search/california,+USA

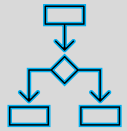
Research Question: How to efficiently schedule these requests?

2

Research Approach

OUR GOAL

To efficiently schedule requests in a UAM system



Develop a Markov Decision Process (MDP) model of the UAM system

Use existing MDP-solving algorithms to find the optimal scheduler

2

Literature Review

Gilles Geeraerts et al. (2018)	Damien Busatto-Gaston et al. (2021)
Define a scheduler for preemptible[1] **tasks** and **jobs** and outlines how to build **a finite MDP** that **accurately represents** the system	The authors discuss a safe **sampling algorithm** and prove that sample complexity is proportional to the size of the domain of the distribution.
Outline a method for finding a **safe and optimal scheduler** for the **MDP model.**	Show that **Monte-Carlo Tree Search (MCTS)** can be used to solve **preemptible[1]** scheduling systems with large state-spaces (e.g. 10^{20})

[1]Preemptible jobs can be interrupted and resumed. Non-Preemptible jobs must be completed before a new job can be started (Buttazzo et al. (2012))

The MDP model, sampling algorithm, and MCTS used by these authors served as a foundation for our approach

Developing MDP model of the UAM System

UAM Scheduler System

- **Goal**: Model requests across a finite set of vertihubs
- **Routes:** Route between two vertihubs. Used to generate requests (C_i, D_i, A_i)
- **Requests:** Specific occurrence of travel on the route
- **Elements of Request:**
  - C_i : The completion time of the request
  - D_i : The deadline for the request (Hard or Soft)
  - A_i : The inter-arrival time for a new request on the same route
- Ex. ({3: 0.5, 4: 0.5}, 7, {8: 1.0})
- $C_i \leq D_i < A_i$

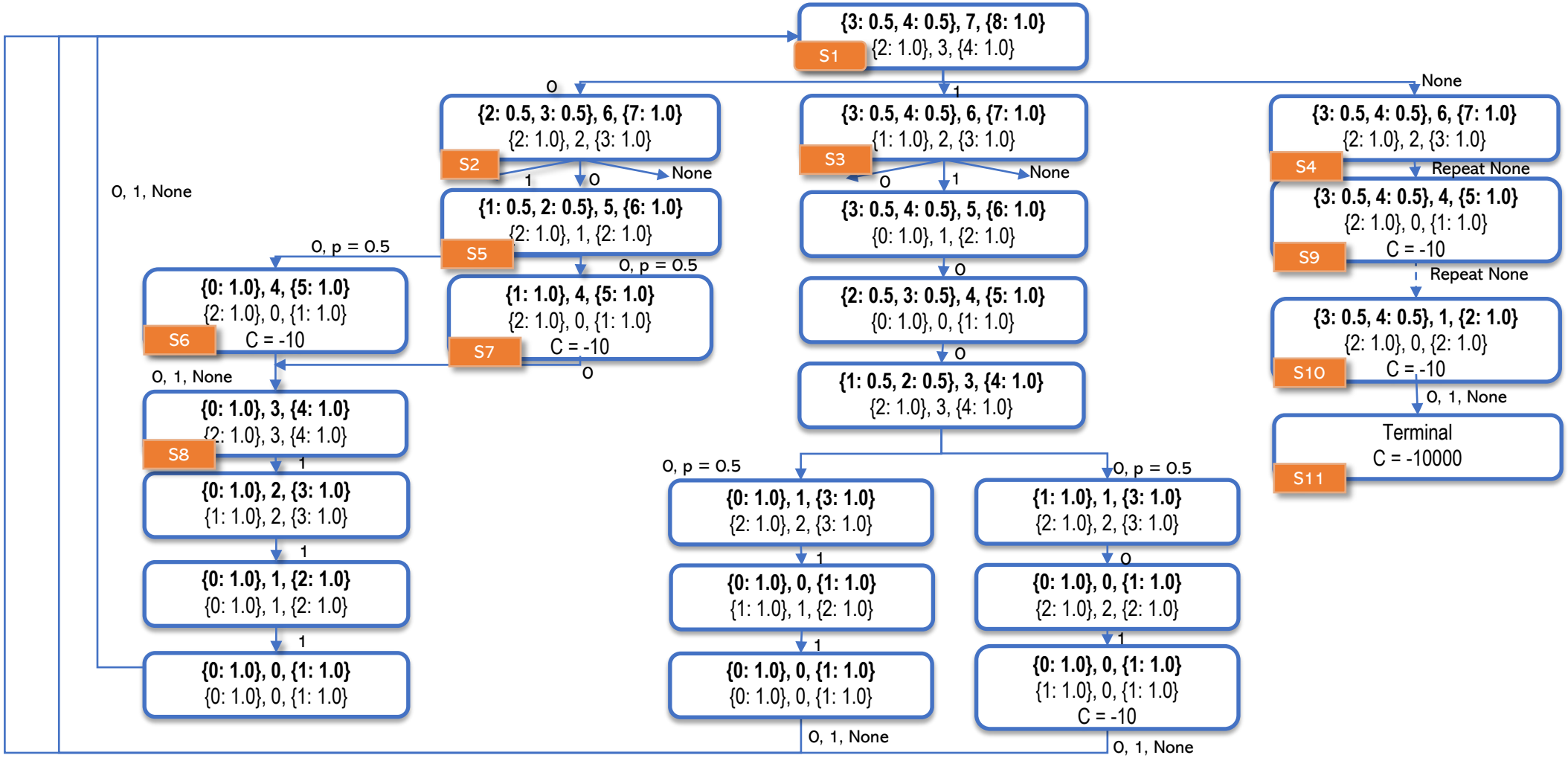
3

UAM Scheduler to MDP

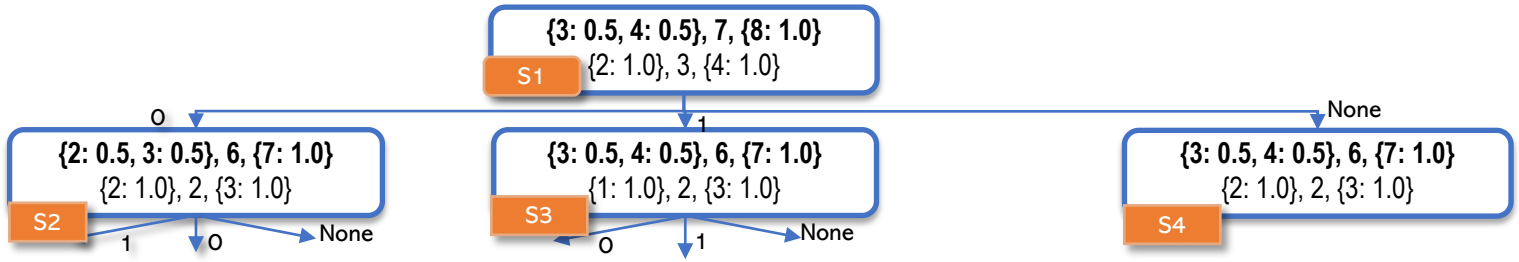
- MDP is widely-used way to model semi-random decision-making processes
- Many currently existing learning algorithms work well with MDPs

MDP	UAM Scheduler
Set of states: S	All reachable request configurations
Set of actions: A	Set of requests an agent can work on
Probability function: P(s, a, s’)	Dependent on the completion and inter-arrival probability distributions
Reward function: R(s, a, s’)	Penalize missed deadlines

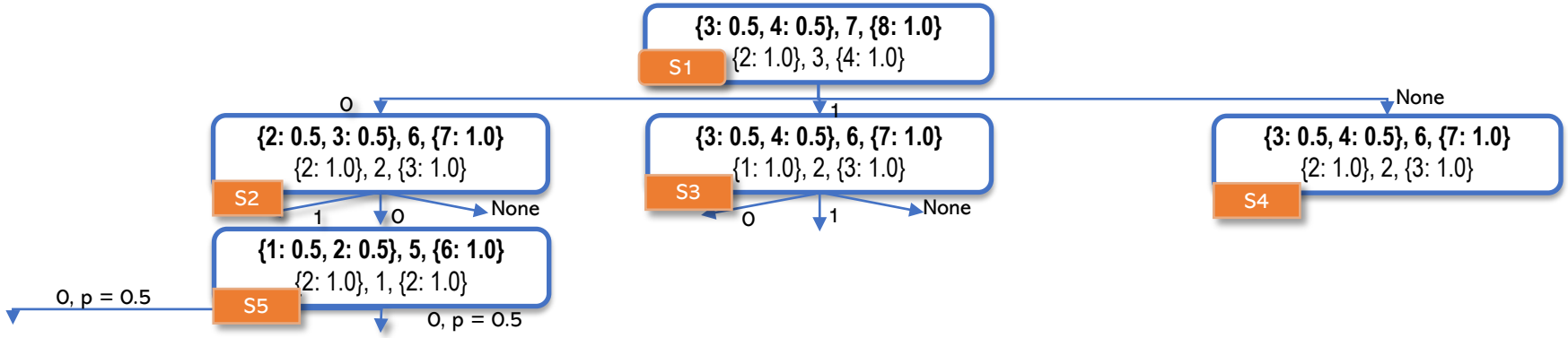
3 Using Basic MDP to illustrate a general task system



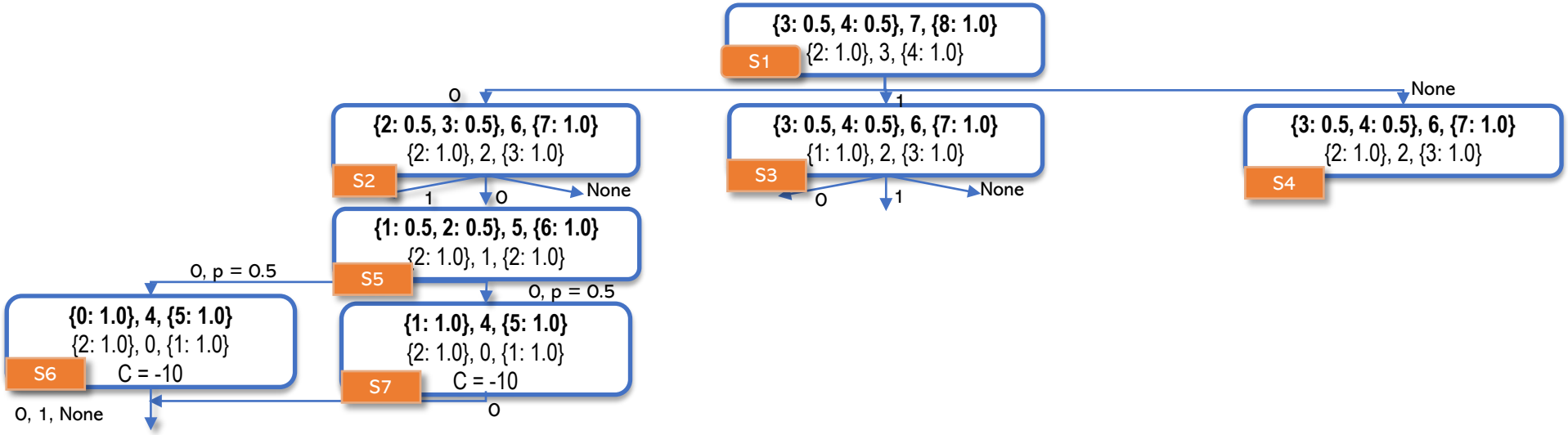
3 Using Basic MDP to illustrate a general task system



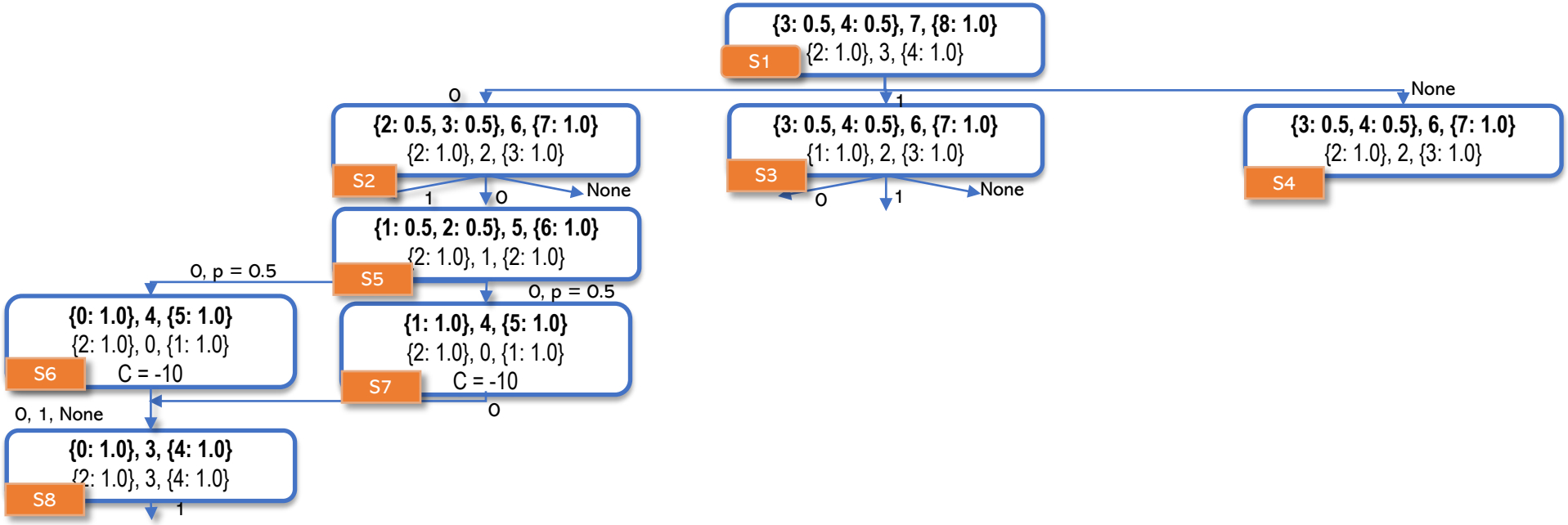
3 Using Basic MDP to illustrate a general task system



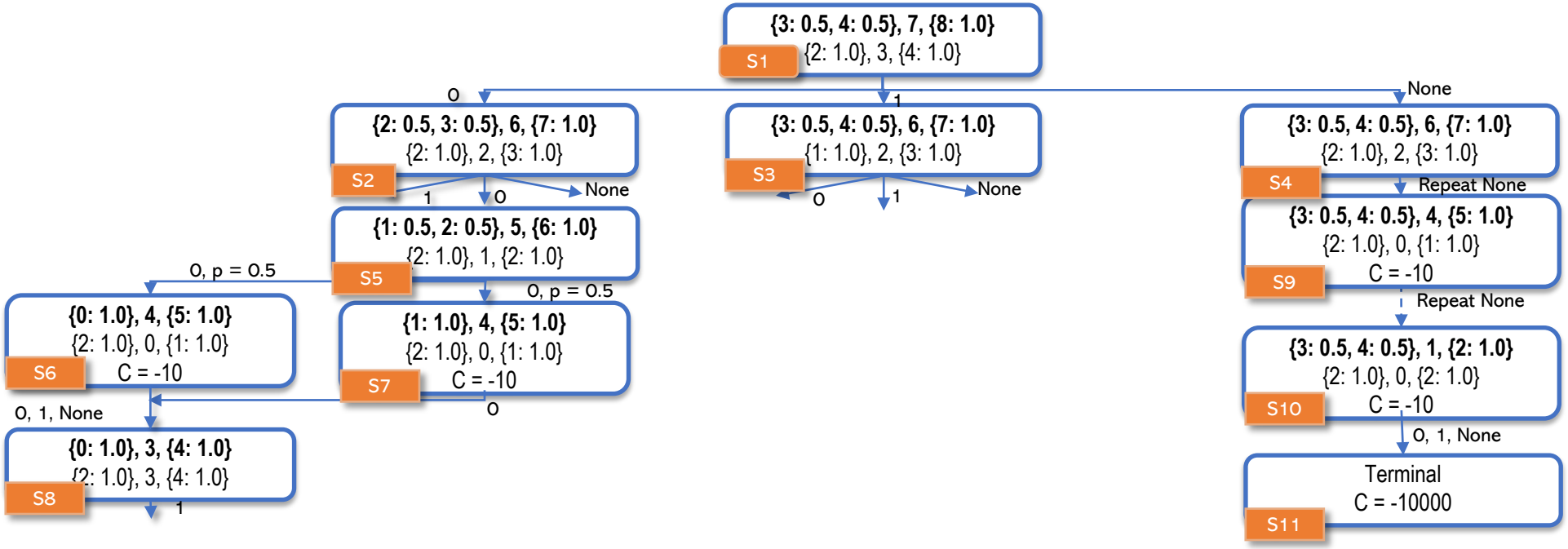
3 Using Basic MDP to illustrate a general task system



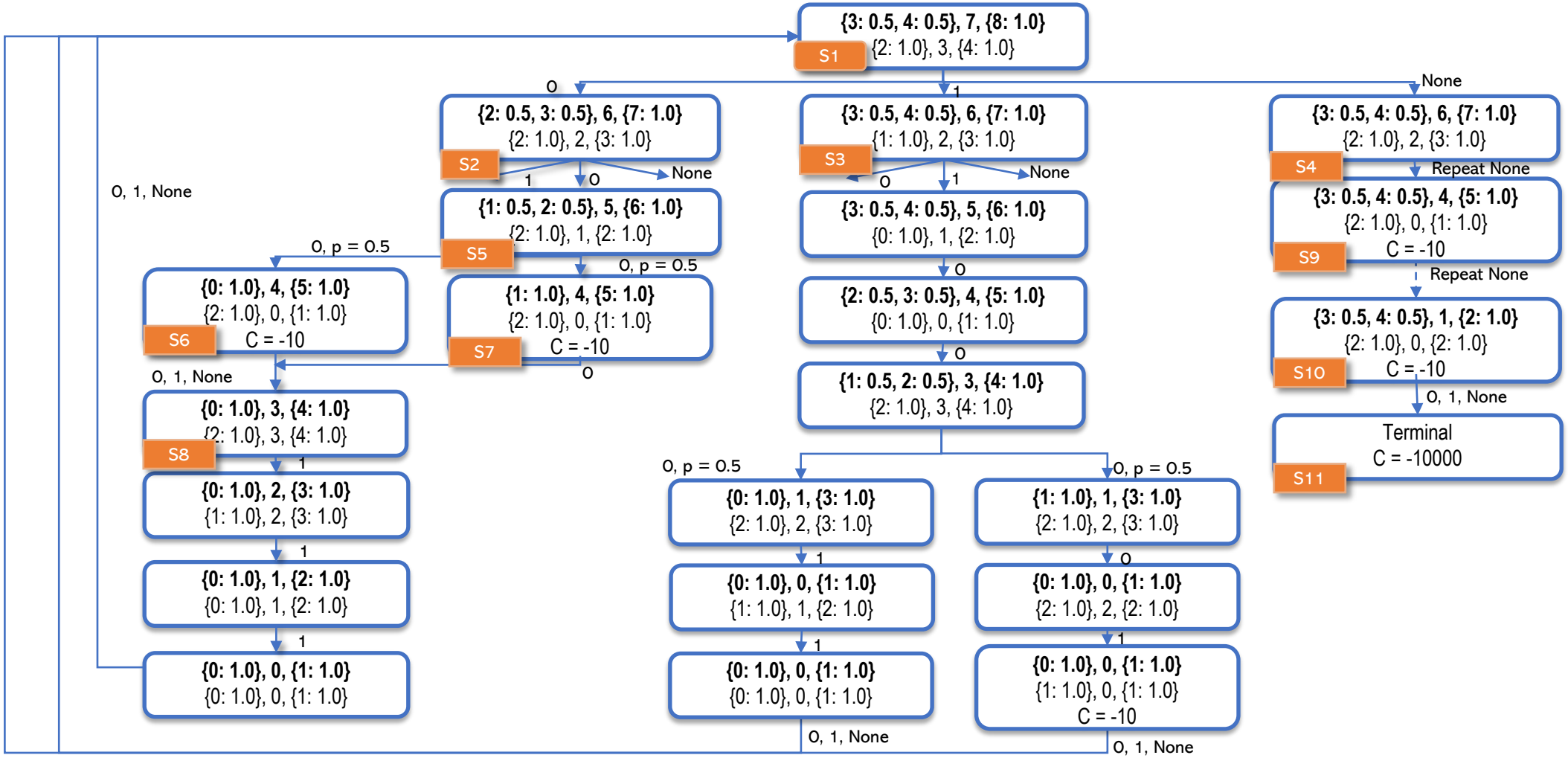
3 Using Basic MDP to illustrate a general task system



3 Using Basic MDP to illustrate a general task system



3 Using Basic MDP to illustrate a general task system



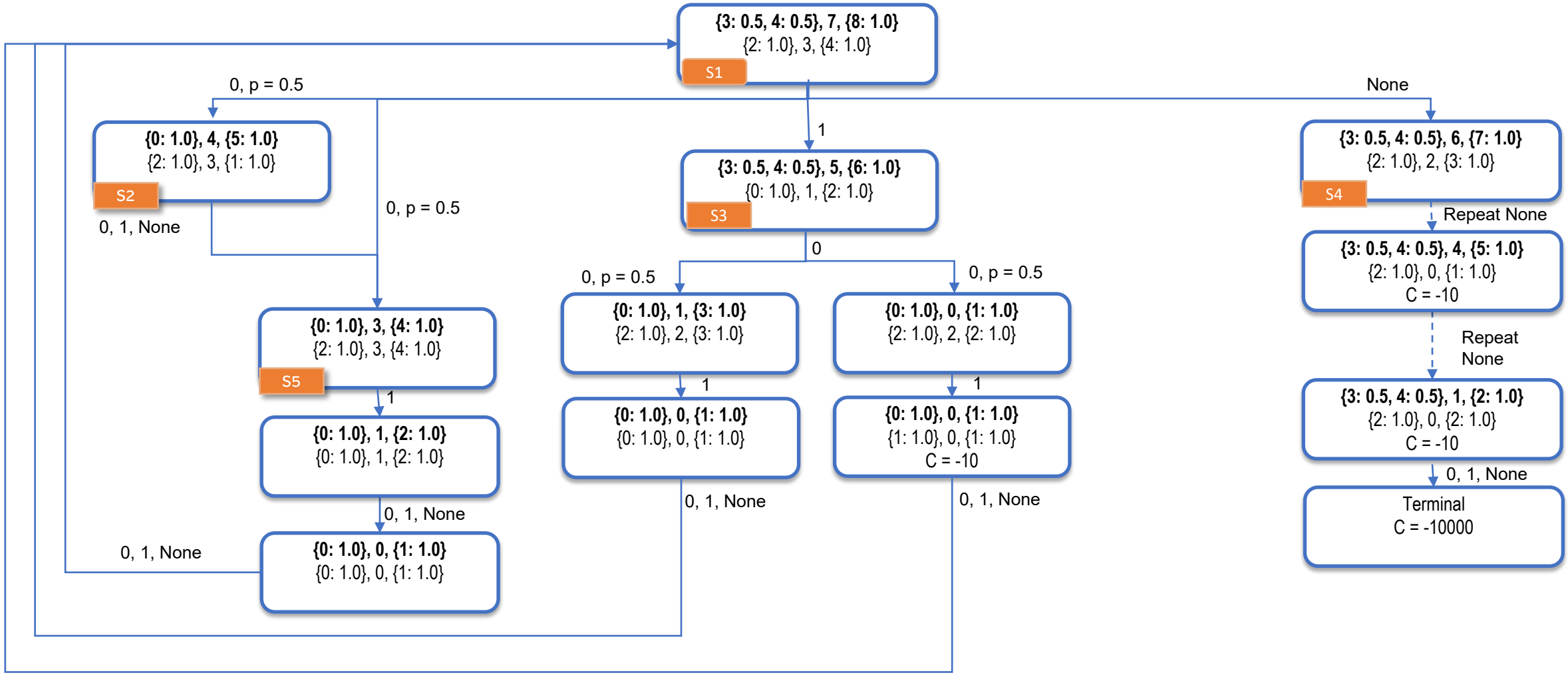
3

Non-Preemptible Model simulates real-world schedule scenarios

- The basic model fails to satisfy the key constraints of a UAM scheduling system
- Since the requests can be preempted, a scheduling agent can stop a request before it is complete.
- This is solved by introducing the non-preemptible constraint on the MDP
- Once a scheduling agent starts a request, it must complete the request before starting a new request
- If the request is replaced before it can be completed the agent is free to choose a different request to work on

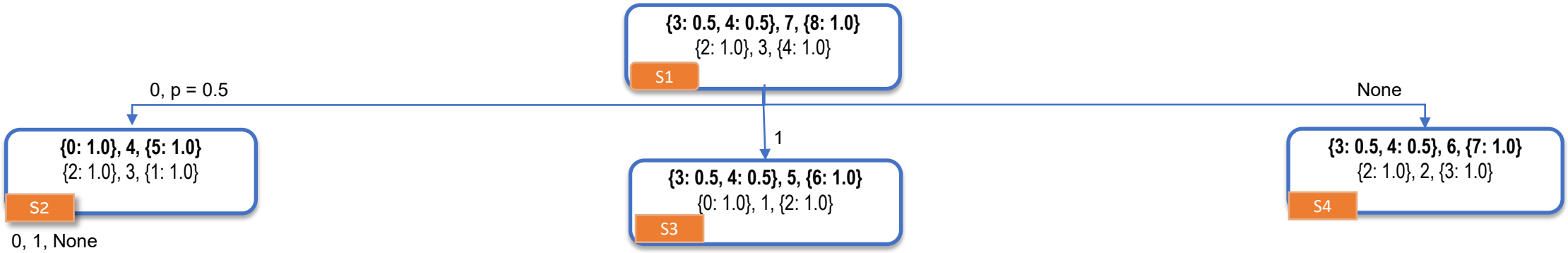
3

Non-Preemptible MDP improves the scheduling approach for UAM



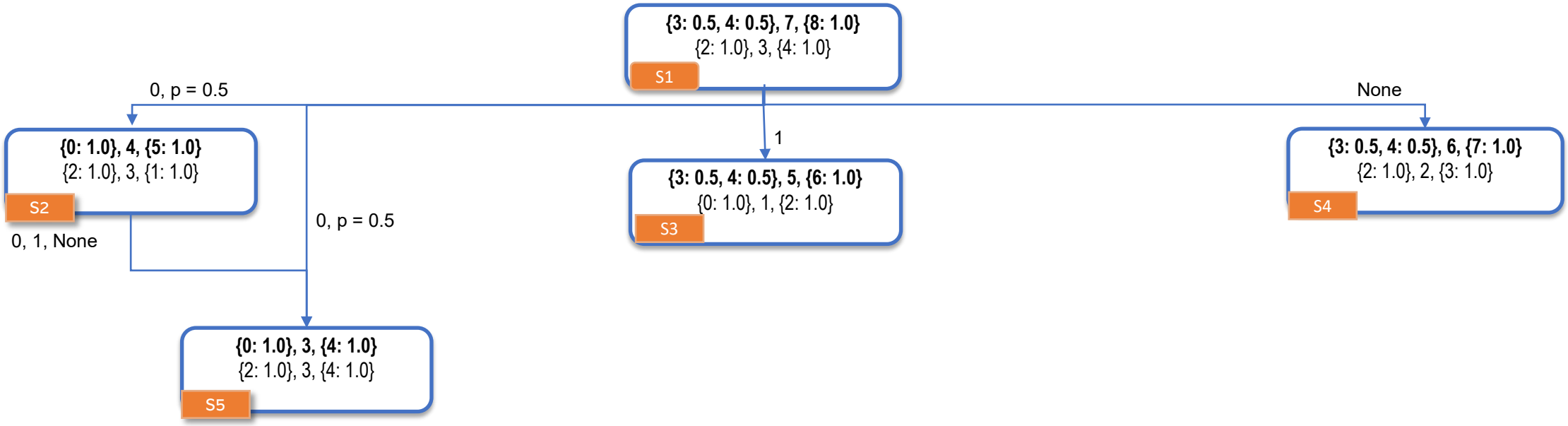
3

Non-Preemptible MDP improves the scheduling approach for UAM



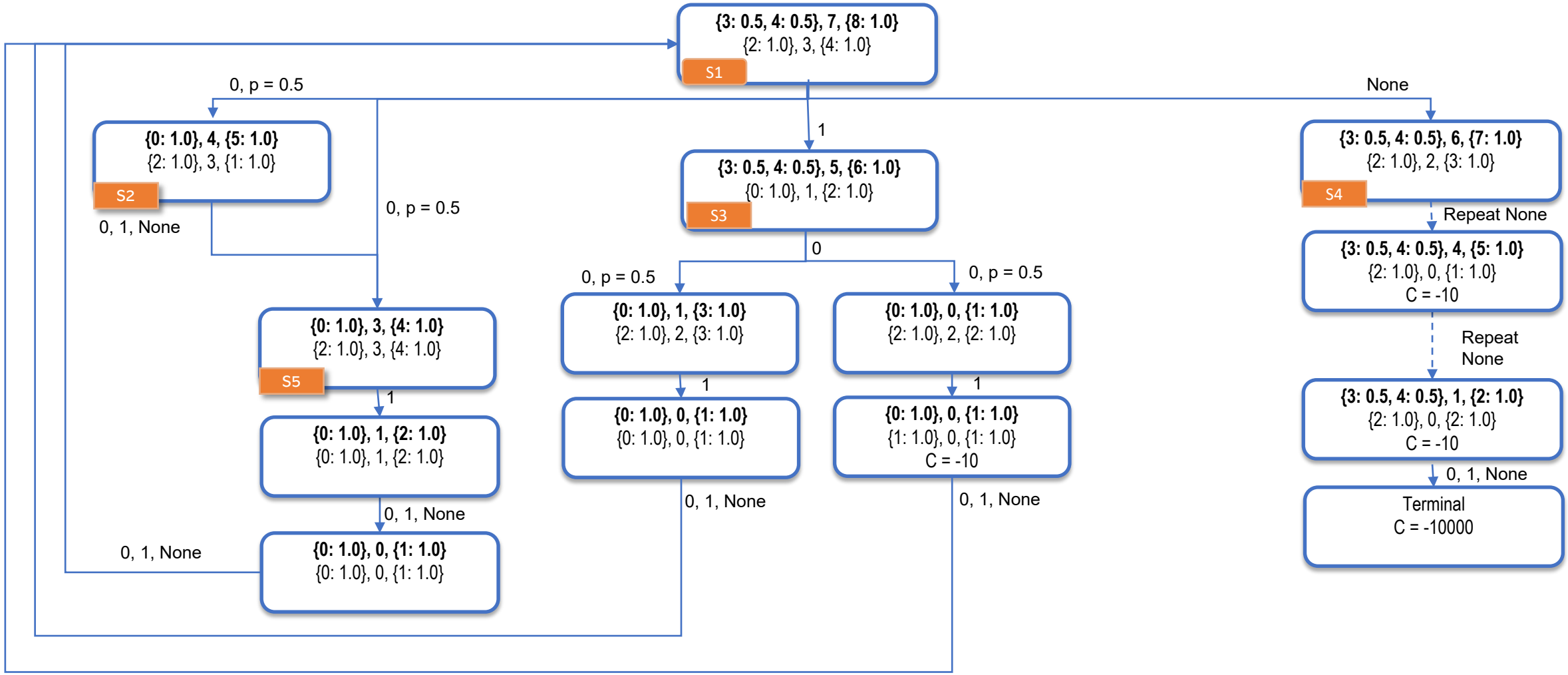
3

Non-Preemptible MDP improves the scheduling approach for UAM



3

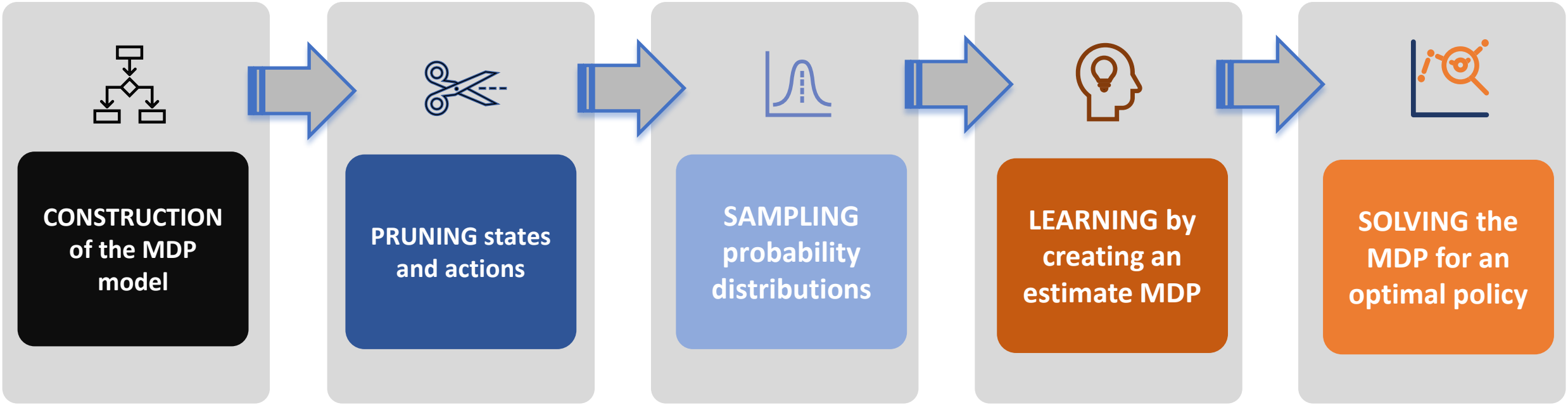
Non-Preemptible MDP improves the scheduling approach for UAM



Solving the MDP

4

The 5-step MDP Process yields optimal policy for UAM Scheduler



MDP Construction and Scalability

The size of the MDP state space depends on the number of requests and the complexity of the completion time and inter-arrival time distributions

# Routes	Preemptible # States	Non-Preemptible # States
1H & 1S	47	18
1H & 2S	82	23
1H & 3S	131	28

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

Completion Time (Delay) Values	P(Delay)	Preemptible # States	Non-Preemptible # States
$\{3\}$	$p_3 = 1$	47	18
$\{3,4\}$	$p_{3\_4} = 0.5$	54	18
$\{1,2,3,4\}$	$p_{1\_4} = 0.25$	59	21

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

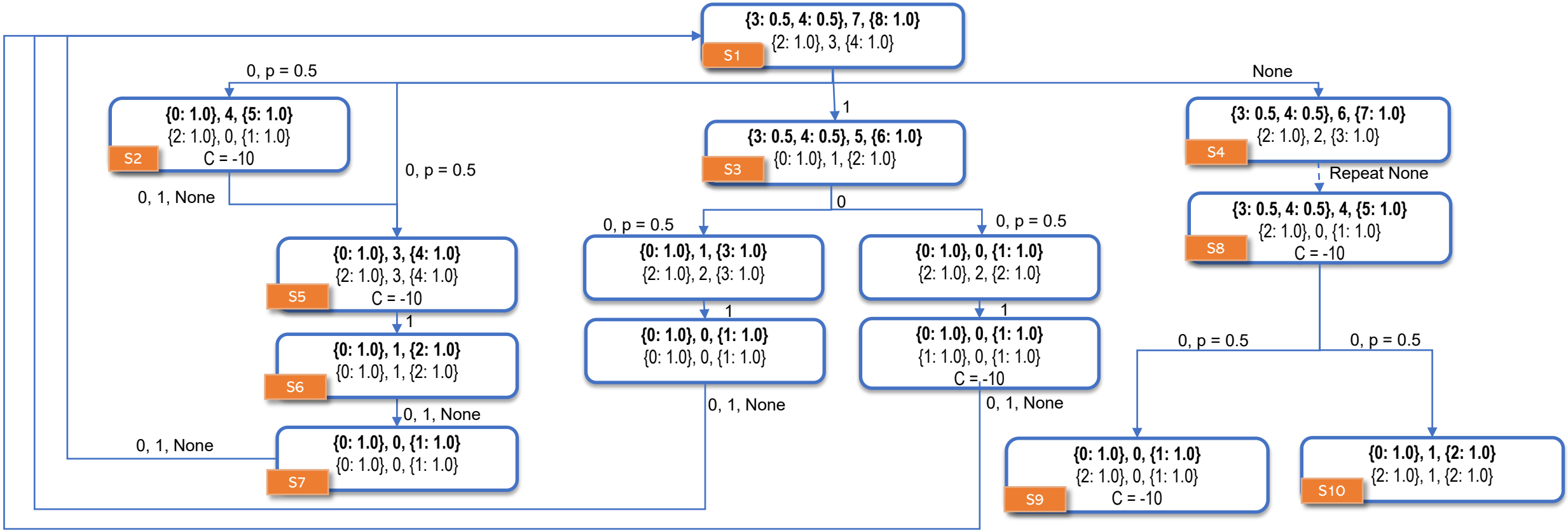
Inter-Arrival Time (Demand) Values	P(Demand)	Preemptible # States	Non-Preemptible # States
$\{8\}$	P = 1	47	18
$\{8,9\}$	P = 0.5	201	75
$\{8,9,10,11\}$	P = 0.25	219	87

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

Pruning the MDP

Key Question: How to avoid entering the terminal state while sampling

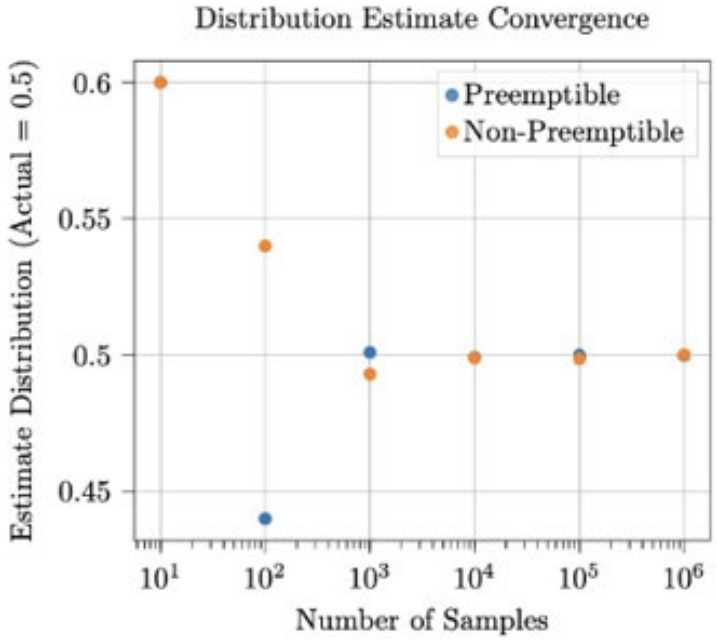
Solution: Prune the MDP to remove terminal states and actions that lead to the terminal state



Sampling Algorithm

- Steps through MDP
- Counts computation time and inter-arrival time
- One can observe how the MDP converges on the actual value as the number of samples increases

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



Solving: Value Iteration

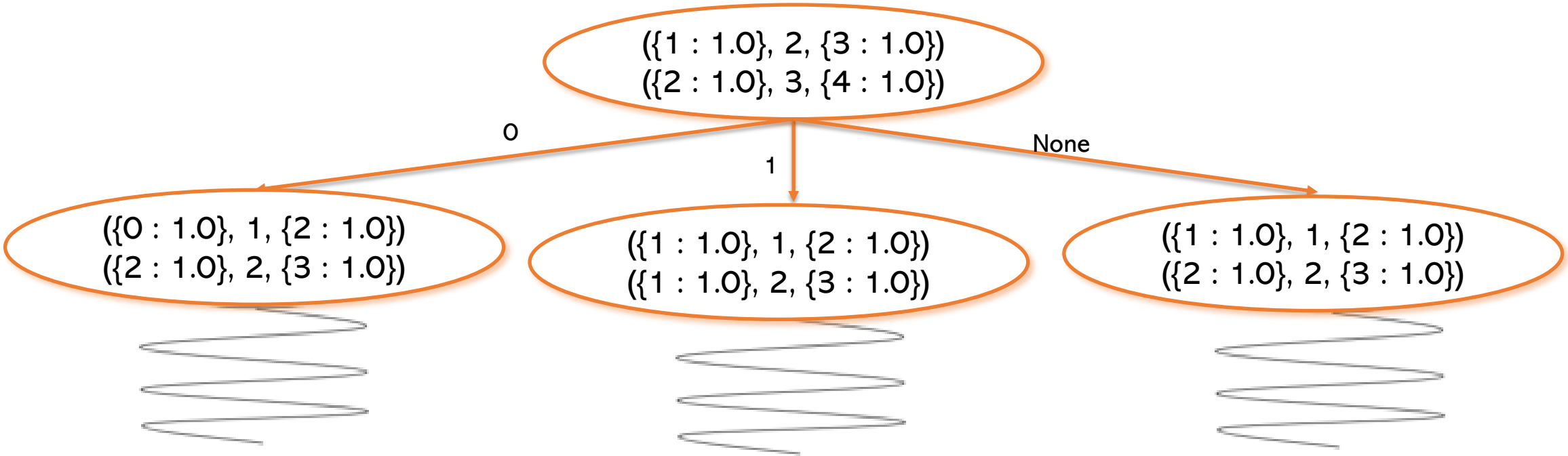
- The value iteration algorithm is used to estimate the discounted future rewards for all states in an MDP.
- The algorithm iteratively updates the values of each state until the discounted future reward converges.
- The run time of value iteration is proportional to the size of the state space, which causes the runtime to increase drastically as the number of requests increases.

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

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

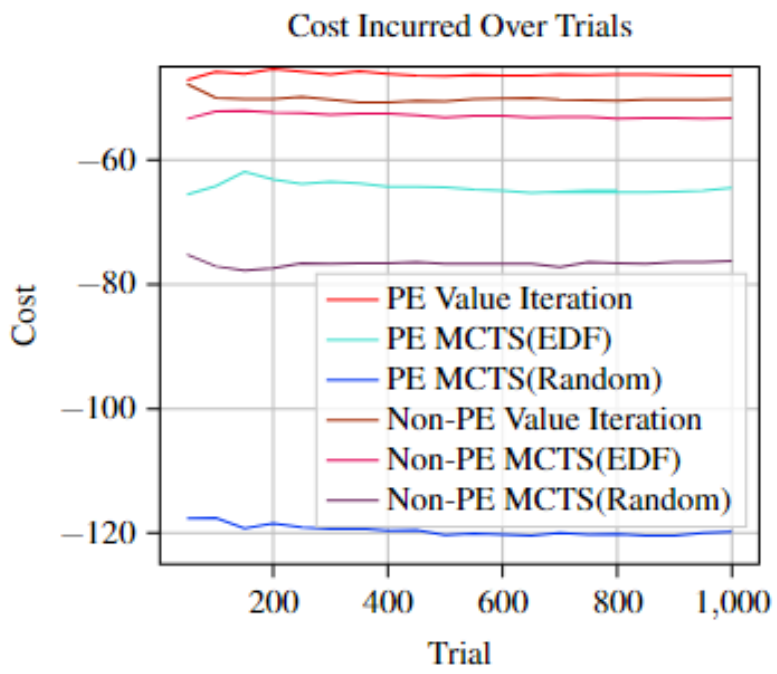
Solving: Monte-Carlo Tree Search

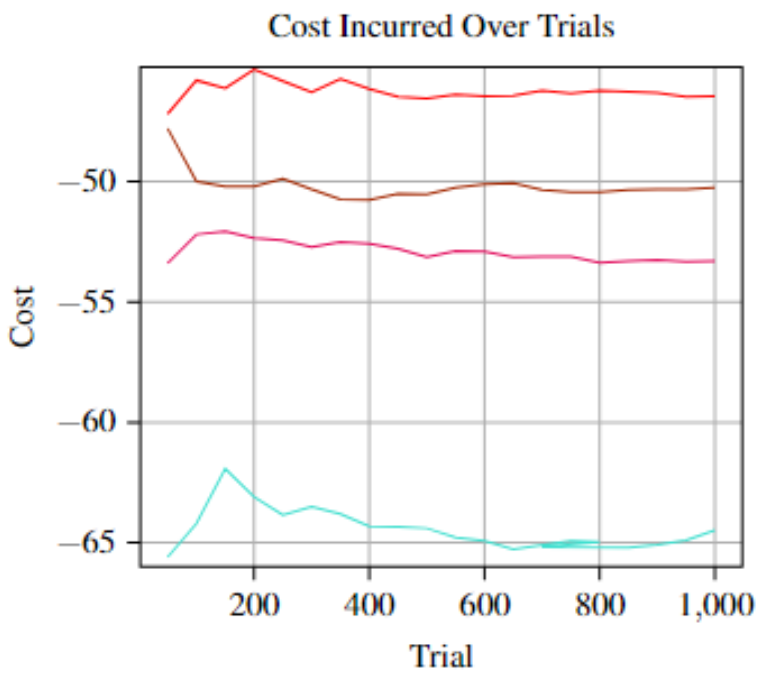
- MCTS is a Finite Horizon Simulation.
- Returns estimated optimal action for a given state
- Since the estimation does not occur for every state, using MCTS greatly reduces run time
- There exist several options for simulation step policies: Random Action vs Earliest Deadline First



Solving: Policy Comparison

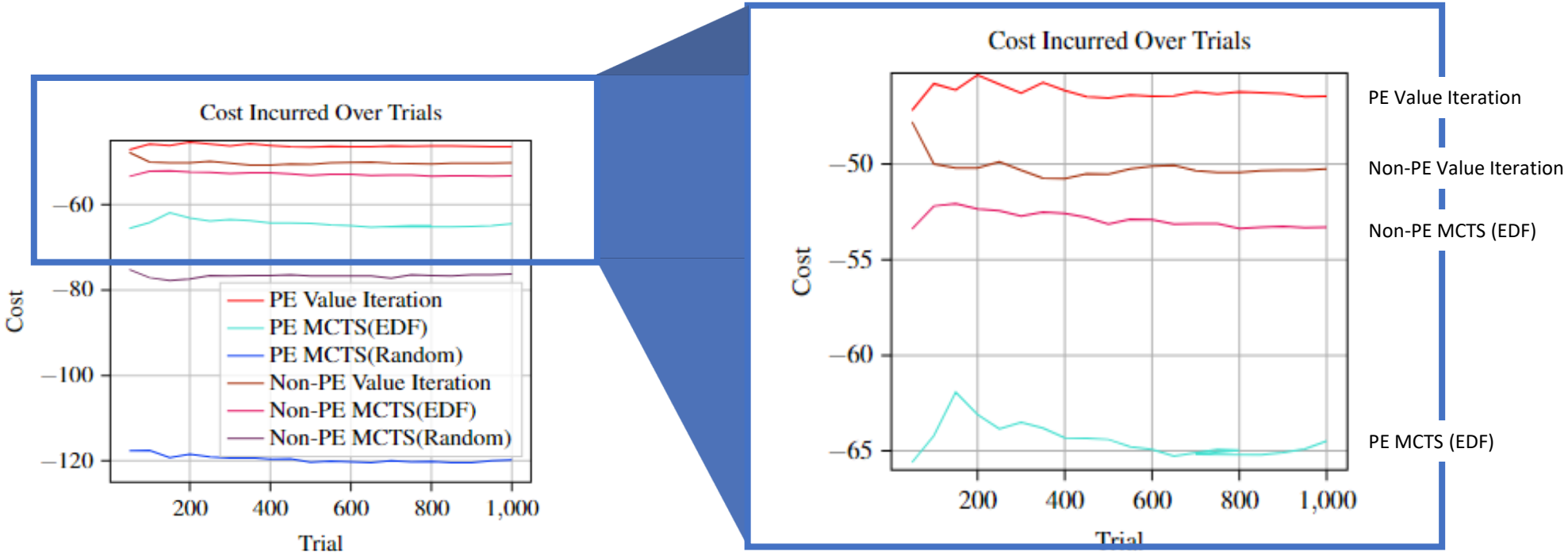
- EDF MCTS performed better than Random MCTS
- Value Iteration performed better than MCTS in all cases.
- Since the non-preemptible scheduler is less flexible than the preemptible, its value iteration policy accrues a larger cost over time
- For MCTS, the non-preemptible scheduler performed better because it was able to step further into the future compared to the preemptible scheduler.





Solving: Policy Comparison

- Value Iteration performed better than MCTS in all cases.
- EDF MCTS performed better than Random MCTS
- Since the non-preemptible scheduler is less flexible than the preemptible, its optimal policy accrues a larger cost over time
- For MCTS, the non-preemptible scheduler performed better because it was able to step further into the future compared to the preemptible scheduler.



Conclusions and Future Work

5

Conclusions and Future Work

- We were able to map a UAM scheduling system to an MDP model.
- By solving the model, we are finding an optimal scheduler for the UAM System

- We were able to synthesize an estimate MDP using the sampling algorithm.

- The 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 the number of routes and size of both computation and inter-arrival distributions

Future Work

Expanding the scheduling framework to multi-agent systems

Introducing additional constraints to improve the accuracy of the model

Questions?

credits: NASA/Lillian Gipson

More Information: Sampling

- Equation used to determine the number of samples:
  - $y = r * \left\lceil \frac{1}{2\varepsilon^2} (\ln(2r) - \ln(\gamma) \right\rceil$
  - r is the size of the domain of the probability function being estimated
  - $1 - \gamma$ is the probability that the estimated distribution is within ε of the actual distribution
- Steps of execution:
  - $s = |F| * A_{max} * D \left\lceil \frac{1}{2\varepsilon^2} (\ln(4D|F|) - \ln(\gamma) \right\rceil$
  - F is the set of routes/tasks in the system
  - A_{max} is the maximum inter-arrival time
  - D is the size of the largest inter-arrival time domain
- This is discussed in more detail in Damien Busatto-Gaston et al. (2021)