Accelerated Structure-Aware Reinforcement Learning for Delay-Resource-Aware Flow Allocation in Wireless Systems

Mahesh Ganesh Bhat, Shana Moothedath, *Senior Member, IEEE,* and Prasanna Chaporkar

***Abstract*—This paper develops a structure-aware reinforcement learning (RL) approach for delay- and energy-aware flow allocation in resource-constrained 5G User Plane Functions (UPFs). We consider a dynamic system with K heterogeneous UPFs of varying capacities that handle stochastic arrivals of M flow types, each with distinct rate requirements. We model the system as a Markov decision process (MDP). We propose a post-decision state (PDS)-based learning approach that exploits the underlying structure of the MDP. By decoupling action-controlled dynamics from exogenous factors and exploiting the monotonic structure of the value function, PDS facilitates faster convergence and efficient adaptive flow allocation in the absence of statistical knowledge of the exogenous variables. We propose two PDS-based algorithms: (i) PDS-value iteration and (ii) PDS-Q learning algorithms. We prove the convergence of the algorithms and compare their memory and computational requirements. Although PDS-based learning enables faster convergence when compared to standard learning approaches such as Q-learning, the exponential growth of the state space with increasing problem size causes even PDS methods to suffer from the curse of dimensionality. To address this, we introduce a tournament-based approximation approach that decomposes the K-UPF problem into multiple 2-UPF sub-problems, each solved using the PDS learning method and then combine the sub-problem policies to obtain the policy of the original problem. This decomposition enables a scalable framework to efficiently approximate the solution for larger instances. Simulation results demonstrate that the proposed PDS algorithms converge faster and achieve a lower long-term cost than standard Q-learning, highlighting the effectiveness of PDS-based RL for resource allocation in wireless networks. We compared the tournament approach against standard benchmarks and a greedy strategy, and validated its performance through empirical evaluation.**

***Index Terms*—Reinforcement learning, resource allocation, UPF flow allocation, energy-aware decision-making**

I. Introduction

The evolution of modern 5G and beyond networks (B5G) has enabled a wide range of applications with diverse service requirements, including high-throughput broadband, latency-critical control applications, and large-scale IoT connectivity. This has led to increasing network traffic with heterogeneous and stringent quality of service (QoS) requirements. The rapid growth of infrastructure to support large amounts of traffic has led to an increase in the energy footprint. As a result, sustainability is a primary objective in the design of modern networks. The recommendation ITU-R M.2160 identified sustainability as one of the highlights in the clauses of "*Motivation and societal considerations*" and "*User and application trends*". This motivates energy-aware resource management and greener network deployments. With this objective in focus, while extensive work has addressed resource allocation and scheduling in Radio Access Networks (RAN) [1], [2], it is important to consider the same in the User Plane Function (UPF), which plays a pivotal role in the 5G core.

The UPF acts as a central data-plane interconnection point within the 5G core, responsible for data forwarding, traffic routing, and enforcing QoS policies. Traffic flow allocation to UPFs directly affects network performance and reliability. Furthermore, UPFs are virtual functions deployed on a variety of infrastructures and vary in characteristics such as geographical location, computational capacity, user proximity, and energy profiles (the amount of operational energy contribution from green energy). These factors influence the selection of the UPF and underscore the need for a principled approach for flow allocation, rather than a simple data forwarding approach. Designing intelligent and adaptive flow allocation strategies is essential for achieving scalable, delay-sensitive, and resource-efficient performance in next-generation mobile networks [3].

Reinforcement learning (RL) presents a promising framework for addressing the challenge of flow allocation by enabling data-driven, adaptive decision making in dynamic and uncertain network environments with limited knowledge [4], [5]. Unlike traditional methods, RL can learn optimal policies through interaction with the system, accounting for stochastic arrivals, departures, and system constraints, making it particularly well-suited for dynamic, delay-sensitive and resource-constrained applications in next-generation networks.

This paper develops an RL-based framework for dynamic flow allocation in 5G networks with multiple UPFs and heterogeneous flows. Under the stochastic arrivals and departures, we seek to allocate the incoming flows to specific UPFs while minimizing the delay and power usage in the network. By formulating the problem as a Markov Decision Process (MDP), we capture the stochastic nature of the network traffic and propose novel RL algorithms to learn optimal allocation policies when the transition dynamics of MDP is unknown. Unlike prior works that adopt deep RL–based approximations, we directly solve the MDP using a value-iteration–based algorithm with provable convergence to the optimal solution. To solve the MDP, we propose a reinforcement learning algorithm that exploits the underlying problem structure, including monotonicity of the value function and the separability of the variables. By incorporating a post-decision state (PDS) analysis that separates the endogenous and exogenous dynamics and the monotonicity of value function, which we prove in the paper, the proposed algorithm achieves faster convergence compared to the standard RL approaches. The proposed ap-

M. G. Bhat and S. Moothedath are with Electrical and Computer Engineering, Iowa State University. Email: {mgbhat, mshana}@iastate.edu. P. Chaporkar is with Electrical Engineering, Indian Institute of Technology Bombay. Email: chaporkar@ee.iitb.ac.in

This work is supported by NSF-CNS 2415213, US and MeitY, India.

proach enables real-time, adaptive decision-making, addressing the limitations of static or model-based techniques in highly dynamic environments to optimize performance while ensuring optimal convergence and resource constraints. To the best of our knowledge, the proposed approach constitutes the first framework for energy- and delay-aware flow allocation with provable optimal convergence, in contrast to prior deep RL approaches that lack optimality guarantees [3], [6], [7].

A major challenge that limits the application of RL in real-world systems is scalability. In many domains, the state space grows exponentially with the size of the problem, and the UPF allocation problem studied in this paper is no exception. As the number of UPFs increases, the state space grows exponentially, resulting in the well-known *curse of dimensionality*. To address this challenge in the UPF allocation problem, we propose an approximate approach using a sub-problem decomposition strategy. Specifically, we decompose the original K-UPF problem into individual 2-UPF subproblems, solve each subproblem independently, and then aggregate their solutions to construct a policy for the original problem. We analyzed the performances of our proposed approaches to evaluate their effectiveness, compared them against benchmark approaches, and validated their superior performances and scalability.

The key contributions of this paper are fourfold.

- We propose a dynamic model for the heterogeneous flow allocation problem, considering random flow arrivals and departures, as well as resource-limited UPFs. We formulated the problem as a delay- and energy-aware MDP, enabling principled analysis to learn optimal adaptive allocation policies under unknown stochastic network dynamics and unknown individual UPF costs.
- We propose post-decision state (PDS)-based RL algorithms, a PDS value iteration (PDS-VI) algorithm, and a PDS-Q learning algorithm to solve the dynamic learning problem. The proposed algorithms leverage the problem structure, specifically, the separation between exogenous and controllable dynamics and the monotonicity of the MDP value function, to simplify learning, enhance sample efficiency, and accelerate convergence. We provide convergence guarantees for the proposed PDS algorithms.
- To address scalability in large-scale problems, we propose a tournament-based approximate PDS approach using problem decomposition. The proposed heuristic decomposes the K-UPF problem into multiple 2-UPF subproblems, each solved using the PDS-VI method. The solutions are then combined using a tournament approach to obtain a policy for the original problem.
- We evaluated the performance of the proposed approach via numerical simulation and compared it against the baseline algorithms. For small-scale problems where the optimal solution is computable, we compared the two proposed PDS algorithms with the standard Q-learning algorithm using exact error metrics. For large-scale problems, we evaluated the proposed tournament-based approach against a greedy heuristic and demonstrated its significant performance improvements.

The rest of the paper is organized as follows. In Section II we present the related work. In Section III we present the notations and preliminaries. In Section IV we present the problem formulation and the MDP modeling of the UPF problem. In Section V we present the proposed PDS RL algorithms. In Section VI we present a new approximate algorithm based on the PDS analysis approach. In Section VII we present the simulation analysis and comparison with benchmark algorithms. In Sections VIII, IX we present the theoretical proofs of our results. Finally, in Section X we present the conclusion.

II. Related Work

The flow allocation problem is related to the task scheduling problem and has been extensively studied in the context of computing and communication systems. Earlier works primarily addressed static (optimization) allocation problems [8]–[10], limiting adaptability in dynamic and uncertain environments. Many early works on dynamic scheduling focus on optimal scheduling of computational tasks in the context of cluster and heterogeneous computing [11], [12]. The extension of heterogeneous computing as cloud and edge computing into communication networks necessitates scheduling algorithms with energy and delay awareness [13]. This includes the computational offloading problem to edge/cloud server with energy and delay considerations. A Mobile Edge Computing (MEC) offload scheduling problem with random arrivals is studied and an integrated solution with RL and stochastic gradient descent (SGD) is proposed for a single edge cloud setting [14]. A Deep Reinforcement Learning (DRL) approach is proposed for delay-resource-aware service optimization in satellite-deployed UPFs [3]. DRL-based resource allocation schemes are proposed for minimizing the delay in MEC [6], [7]. DRL approaches rely on approximate function representations and often require extensive training data. Further, in most of these studies, the task arrivals and departures are assumed to be fixed and known. However, the task arrivals and departures are often exogenous and unknown. Moreover, introducing ML-based controller into the network for efficient flow allocation, while intended to improve energy efficiency, may itself incur additional energy costs. In particular, deep learning–based techniques typically require tuning a large number of model parameters, and the energy savings achieved through improved allocation can be offset by the additional energy required to train and update the model. Our goal in this paper is to develop an efficient *reinforcement learning* framework that offers fast and sample-efficient learning by leveraging the problem structure. Another key advantage of our online RL approach is that our proposed online learning algorithm provides a low-complexity solution that ensures the intended energy savings are indeed realized.

The inherent structural properties of allocation and scheduling problems enable post-decision state (PDS)-based RL analysis. PDS analysis has been applied in scheduling and allocation problems. PDS has been studied in the context of delay-sensitive wireless transmission scheduling for energy-efficient point-to-point communication and accelerated learning [15], [16]. However, they mostly focus on packet transmission from a single queue. A PDS-based online learning for server allocation in data centers was introduced in [17] to reduce electricity

costs. Wireless resource scheduling in virtualized RAN networks with random arrivals and departures, focusing on mobile device utility, is considered in [18]. In [18], the spectrum allocation problem is formulated as a non-cooperative stochastic game with the objective of maximizing provider profits. An approximate approach is proposed by decomposing the resulting multi-agent MDPs into multiple single-agent MDPs, and a PDS-based online localized algorithm is developed. RL and DRL-based privacy-aware offloading methods for IoT applications using PDS are studied in [19], [20].

In contrast to these works, we address the sustainability of core network infrastructure by jointly considering energy efficiency and delay in flow allocation. We focus on modeling the dynamic allocation of heterogeneous flow types, under unknown flow arrivals and departures. We aim to leverage the structural properties to learn the optimal policy using PDS-based RL to solve delay- and energy-aware flow allocation, while considering the capacity constraints. To the best of our knowledge, this problem in a multi-server scenario with stochastic arrivals and departures has not been previously studied except for the conference version of this paper [21]. In the conference version, we provided the PDS-VI algorithm and its empirical evaluation. In this journal version, we present a new PDS-based Q learning algorithm. Further, we propose an approximate PDS approach via problem decomposition approach to tackle large-scale systems. We also performed extensive simulations, robustness analysis, and comparison across multiple baseline algorithms.

III. Notations and Preliminaries

A. Markov Decision Process (MDP) Preliminaries

MDP is the standard framework for modeling a stochastic dynamical system and computing its optimal control policy [22], [23]. Mathematically an MDP can be defined as follows. Let $\mathcal{S}$ and $\mathcal{A}$ be compact sets describing the states and actions of the controller (agent), respectively, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The system dynamics is characterized by the probability transition structure $\mathbb{P}$, where $\mathbb{P}(s'|s,a)$ is the probability of transitioning to state s' from state s under control action a. A policy $\pi : \mathcal{S} \to \mathcal{A}$ is a conditional distribution $\pi(a|s)$ that guides the decision-making process of an agent. At time $t \in \{1, 2, \dots, \}$, the agent observes the current state s_t and chooses an action a_t from the policy $\pi(a|s)$, and observes the reward $r(s_t, a_t)$. The action chosen by the agent at a state drives the agent to a next state according to $\mathbb{P}(s_{t+1}|s_t, a_t)$, where s_{t+1} belongs to the set of neighboring states of s_t. A trajectory $\tau = \{s_0, a_0, s_1, a_1, s_2, a_2, \dots, \}$ is a sequence of state-action pairs traversed by the agent following a policy. The goal of the agent is to find an optimal policy $\pi^\star$ that maximizes the cumulative reward $r(\tau) = \lim_{T\to\infty} \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$, where $\gamma \in (0, 1]$ is the discount factor which captures how myopic the agent is. When the model $\mathbb{P}$ is known, $\pi^\star$ can be computed using dynamic programming [22], [23]. However, in many real-world problems, the model is unknown. The goal of a reinforcement learning algorithm is to learn the optimal policy $\pi^\star$ without prior knowledge of the model [23].

IV. Problem Formulation: Delay-Sensitive Flow Allocation in 5G User Plane Functions

In this section, we present the wireless environment (system) and then discuss the modeling and formulation. The key notations are listed in Table I.

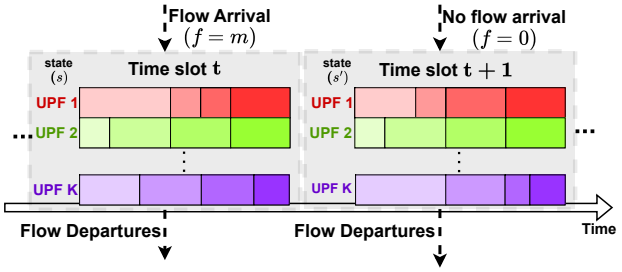


Fig. 1: A schematic representation of the evolution of the time-slotted system. The horizontal blocks represent UPFs. Each cell for a UPF represents a type of flow, and the width of the cell represents the number of flows currently being served by the UPFs. Thus, there are four flow types, i.e., $M = 4$, and a wider cell indicates that a larger number of flows of that type are being served at the UPF. In the example, a flow arrives at time slot t, whereas no arrival occurs at time slot $t + 1$.

TABLE I: List of Important Notations

Symbol	Description
a, $\boldsymbol{a}$	Action and Action indicator matrix of dimension $K \times M$
f	Flow arrival index, $f \in \{0, 1, \dots, M\}$
q_k	Flow departure probability of UPF k
$\boldsymbol{u}$	Departure matrix of dimension $K \times M$
$s, \boldsymbol{n}$	Pre-decision state and allocation $(K \times M)$ matrices
$\tilde{s}, \tilde{\boldsymbol{n}}$	Post-decision state and allocation $(K \times M)$ matrices
$\mathbb{P}(\boldsymbol{n}'\|s, \boldsymbol{a})$	Transition probability of the state matrix
$\mathbb{P}(\tilde{\boldsymbol{n}}\|s, \boldsymbol{a})$	Action controlled dynamics
$\mathbb{P}(\boldsymbol{n}'\|\tilde{\boldsymbol{n}})$	Departure dynamics
$\mathbb{P}(f')$	Flow arrival probability
$\mathbb{P}^k$	Known probability of the transition dynamics
$\mathbb{P}^u$	Unknown probability of the transition dynamics
V	Value vector
$V^\star$	Optimal value function
$\tilde{V}$	Post-decision state value vector
$\tilde{V}^\star$	Optimal post-decision state value vector
Q	Pre-decision state Q-table
$Q^\star$	Optimal pre-decision state Q-table
$\tilde{Q}$	Post-decision state Q-table
$\tilde{Q}^\star$	Optimal post-decision state Q-table

A. System Model

Consider a 5G network. The time is slotted. At the beginning of each slot, a new flow arrives in the network with probability (w.p.) p. Each flow arriving can be of one of M types. Let b_m denote the probability that an arriving flow is of type m for $m \in [M]$, where $[Z] := \{1, 2, \dots, Z\}$ for any integer Z. We also assume that the flow arrivals and its

type are independent across flows. The flow type indicates the average flow rate requirement. Let $\bar{R}_m$ denote the average rate requirement for the flow of type m. Without loss of generality, $\bar{R}_m < \bar{R}_{m+1}$ for every $m \in [M-1]$. For each flow that arrives, the network must decide whether to accept the flow. Specifically, if the required rate cannot be guaranteed, then admission is denied to the flow; otherwise it is accepted.

If a flow is accepted in the network, then it is assigned a UPF that handles the flow until it departs and provides it the required rate based on its type. We assume that there are K UPFs in the network. Each UPF may have distinct capabilities in terms of the available memory, computational and switching speeds. Based on the capabilities, let C_k denote the maximum data rate (in bits/second) that UPF $k \in [K]$ can support. Finally, at most one flow departs the k^{th} UPF w.p. q_k, for $k \in [K]$, at the end of the slot, independently of flow departures in other slots. The departing flow is equally likely to be any existing flow in the UPF. A flow arriving in a slot can depart in the same slot. We now formulate the dynamic UPF allocation problem as an MDP.

B. Markov Decision Process Formulation

Flow allocation in 5G networks is inherently a dynamic decision-making problem. MDP offers a principled framework for capturing such dynamics and optimizing long-term performance. In this section, we model the delay- and energy-sensitive flow allocation problem in UPFs as an MDP and describe its key components in detail.

1) State Space: We define the state of the system as a tuple containing the allocation matrix $\boldsymbol{n}$ and the flow arrival index f, i.e., $s = (\boldsymbol{n}, f)$. For a state $s = (\boldsymbol{n}, f)$, let $\boldsymbol{n}$ be a $K \times M$ matrix such that the $(k,m)^{\text{th}}$ entry $\boldsymbol{n}_{km}$ denotes the number of type m flows handled by k^{th} UPF at s and $f \in \{0, 1, \ldots, M\}$ denotes the type of flow arrived at s. Here, $f = 0$ means there is no flow arrival at s. Let $\tilde{R}_k(s)$ denote the total average rate that UPF k needs to support in s. Note that

$$\tilde{R}_k(s) = \sum_{m \in [M]} \boldsymbol{n}_{km} \bar{R}_m. \tag{1}$$

Let us define the state space $\mathcal{S}$ as

$$\mathcal{S} = \left\{ s : C_k > \tilde{R}_k(s) \text{ for all } k \in [K], f \in \{0, \ldots, M\} \right\}.$$

Note that $\lfloor C_k / \bar{R}_1 \rfloor$ is the maximum number of flows that UPF k can support at any given time. Thus, $\mathcal{S}$ is a finite set.

2) Action Space and Control Policy: The action set is defined as below. Consider a state $s = (\boldsymbol{n}, f)$. Let $\mathcal{A}(s)$ be the set of feasible actions in s. Then $\mathcal{A}(s) \subseteq \{0, 1, \ldots, K\}$, where $k \in \mathcal{A}(s)$ if $C_k > \tilde{R}_k(s) + \bar{R}_f$, for $k \in [K]$. For $a \in \mathcal{A}(s)$, $a \in [K]$ indicates flow f is allocated to UPF a and $a = 0$ indicates the flow is blocked. The flow f in state s must be admitted as long as $\mathcal{A}(s) \neq \{0\}$ and will be handled by the allocated UPF until it departs. The action set is of dimension $K+1$. For $a \in \mathcal{A}(s)$, we define a $K \times M$ indicator matrix, $\boldsymbol{a}$, for analytical purposes.

$$\boldsymbol{a}_{km} = \begin{cases} 1, & \text{if } a = k \text{ and } f = m, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Thus, $\boldsymbol{a}$ is a sparse indicator matrix with at most one non-zero entry (equal to 1), with all remaining entries being 0s.

A control policy $\pi : \mathcal{S} \to \mathcal{A}$ maps the states to feasible action. We assume that π is causal, i.e., the action chosen for state s may depend on the past states and actions taken, but not on future evolution. Moreover, π can also be a randomized policy, i.e., for a state action can be chosen randomly from the set of feasible actions.

3) Cost Function: Let the cost incurred in state s be $\xi(s)$,

$$\xi(s) = \sum_{k \in [K]} (\alpha_k(s) + \delta_k(s)), \tag{3}$$

where $\alpha_k(s)$ denotes power cost, and $\delta_k(s)$ denotes the delay cost at UPF k in state s. These costs are given as

$$\alpha_k(s) = c_k \tilde{R}_k(s) \text{ and } \delta_k(s) = \frac{C_k}{C_k - \tilde{R}_k(s)}. \tag{4}$$

Here, c_k is power cost for switching one bit at UPF k.

4) System Dynamics: The system dynamics is characterized by the probability transition dynamics $\mathbb{P}$, where $\mathbb{P}(s'|s, \boldsymbol{a})$ is the probability of transitioning to the state s' from the state s under control action $\boldsymbol{a}$. Consider the current state $s = (\boldsymbol{n}, f)$ and the next state $s' = (\boldsymbol{n}', f')$, where $\boldsymbol{n}, \boldsymbol{n}'$ denote the flow allocation matrices and f, f' the flow arrivals.

The transition dynamics consist of two components: (i) the evolution of the allocation matrix $\boldsymbol{n}$, which depends on the current state, control action, and stochastic departures, and (ii) the arrival process f, which is exogenous and independent of the control action, follows a fixed but unknown probability distribution. Formally, the transition probability

$$\mathbb{P}(s'|s, \boldsymbol{a}) = \mathbb{P}(\boldsymbol{n}'|s, \boldsymbol{a}) \cdot \mathbb{P}(f'), \tag{5}$$

where the arrival probability is given by

$$\mathbb{P}(f') = (1-p)\mathbb{1}_{\{f'=0\}} + p \sum_{m=1}^{M} b_m \mathbb{1}_{\{f'=m\}}. \tag{6}$$

The evolution of the allocation matrix $\boldsymbol{n}$ depends on the departure process. To define $\mathbb{P}(\boldsymbol{n}'|s, \boldsymbol{a})$, we first define the departure process. Let $\boldsymbol{n}_k$ and $\boldsymbol{n}'_k$ be the row vectors corresponding to UPF k in the allocation matrices at the current state s and next state s', respectively. Define the canonical vector in $\mathbb{R}^M$ as

$$\boldsymbol{e}_m \in \{0,1\}^M, \qquad (\boldsymbol{e}_m)_j = \begin{cases} 1, & \text{if } j = m, \\ 0, & \text{if } j \neq m \end{cases}$$

Consider a scenario where a flow of type m departs. Let $\mathcal{D}_k(\boldsymbol{n}'_k, \boldsymbol{n}_k, f)$ be the transition probability of UPF k, i.e., probability of transitioning from $\boldsymbol{n}_k$ to $\boldsymbol{n}'_k$ under arrival f. Let us first consider the case where no new flow arrives in s or flow of type $\tilde{m}$ arrives but $\mathcal{A}(s)$ is empty. Then, either $\boldsymbol{n} = \boldsymbol{n}'$, or the allocation matrix transitions to a distinct state due to flow departures. Then,

$$\mathcal{D}_k(\boldsymbol{n}'_k, \boldsymbol{n}_k, f) = \begin{cases} 1 - q_k, & \text{if } \boldsymbol{n}'_k = \boldsymbol{n}_k \\ q_k \cdot \frac{\boldsymbol{n}_{km}}{\sum_{m'} \boldsymbol{n}_{km'}}, & \text{if } \boldsymbol{n}'_k = \boldsymbol{n}_k - \boldsymbol{e}_m \\ 1, & \text{if } \boldsymbol{n}'_k = \boldsymbol{n}_k = \boldsymbol{0} \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Now consider a case where a new arrival of type $\tilde{m}$ arrives at state s and $\mathcal{A}(s)$ is non-empty. Without loss of generality, let $a = k'$. Two cases arise.

Case 1: For any UPF $k \neq k'$. The departure probability in this case is same as in Eq. (7).

Case 2: For UPF k'. Then,

$$\mathcal{D}_{k'}(\boldsymbol{n}'_{k'}, \boldsymbol{n}_{k'}, f) = \begin{cases} 1 - q_{k'}, & \text{if } \boldsymbol{n}'_{k'} = \boldsymbol{n}_{k'} + \boldsymbol{e}_{\tilde{m}} \\ q_{k'} \cdot \frac{\boldsymbol{n}_{k'\tilde{m}}+1}{\sum_{m'} \boldsymbol{n}_{k'm'}+1}, & \text{if } \boldsymbol{n}'_{k'} = \boldsymbol{n}_{k'} \\ q_{k'} \cdot \frac{\boldsymbol{n}_{k'm}}{\sum_{m'} \boldsymbol{n}_{k'm'}+1}, & \text{if } \boldsymbol{n}'_{k'} = \boldsymbol{n}_{k'} + \boldsymbol{e}_{\tilde{m}} \\ & -\boldsymbol{e}_m \text{ and } m \neq \tilde{m} \\ 0, & \text{otherwise.} \end{cases}$$

The transition probability

$$\mathbb{P}(\boldsymbol{n}'|s, \boldsymbol{a}) = \prod_{k=1}^{K} \mathcal{D}_k(\boldsymbol{n}'_k, \boldsymbol{n}_k, f). \tag{8}$$

Eqs. (5), (6), (8) complete the modeling of system dynamics.

5) Optimal Policy: Let $s^\pi(t)$ be the state in time slot t under policy π. For any function $g : \mathcal{S} \to \mathbb{R}$, we use the shorthand $g^\pi(t)$ to denote $g(s^\pi(t))$. For instance, $\tilde{R}^\pi_k(t)$ denotes the total rate that UPF k must support at time t under policy π, i.e., $\tilde{R}_k(s^\pi(t))$, and $\xi^\pi(t)$ denotes the cost incurred at time t, i.e., $\xi(s^\pi(t))$. Our goal is to learn optimal policy $\pi^\star$ that minimizes the cumulative discounted cost

$$\xi^\pi = \lim_{T\to\infty} \sum_{t=0}^{T} \gamma^t \xi^\pi(t).$$

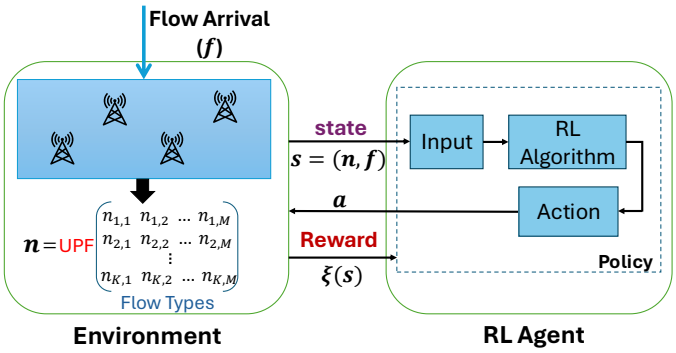


Fig. 2: Schematic of the MDP model and RL approach.

V. Proposed Structure Leveraging Reinforcement Learning Approach

In this section, we present reinforcement learning (RL) algorithms to compute the optimal policy for the system. In practical settings, the arrival probabilities (p, b_m) and the departure probabilities (q_k) are often unknown, rendering classical dynamic programming approaches inapplicable. To address this, we propose model-free RL algorithms that learn the optimal value function through interaction with the environment, thereby enabling the derivation of an optimal policy without requiring prior knowledge of the system dynamics. We begin by introducing the Q-learning algorithm, a widely used method for learning optimal policies in settings where system dynamics are unknown.

A. Preliminaries: Q-Learning

Q-learning is a commonly used learning algorithm for value and policy estimation in models with unknown transition dynamics. In standard Q-learning for cost-minimization, each state-action pair specific value function is estimated such that $V(s) = \min_{\boldsymbol{a}} Q(s, \boldsymbol{a})$, where

$$Q^\pi(s, \boldsymbol{a}) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \xi_{t+1} \middle| s_0, \boldsymbol{a}_0 \right],$$

where s_0 and $\boldsymbol{a}_0$ are the initial state and action, and γ is the discount factor. Q-value is the expected cumulative reward/cost for taking a specific action in a specific state and then following the optimal policy thereafter, and Q-table is the tabular representation of the Q-value function. The core of the conventional Q-learning algorithm is the Q-value update based on the known information tuple $(s, \boldsymbol{a}, \xi, s')$, given by

$$\begin{aligned} Q_{t+1}(s, \boldsymbol{a}) =& Q_t(s, \boldsymbol{a}) + \\ & \alpha_t \big[\xi(s) + \gamma \min_{\boldsymbol{a}'} Q_t(s', \boldsymbol{a}') - Q_t(s, \boldsymbol{a}) \big], \end{aligned} \tag{9}$$

where α_t is the learning rate in the t^{th} iteration of Q updates. While Q-learning is a standard method for RL, the transition structure of the problem enables us to leverage the concept of post-decision state in our analysis.

B. Post-Decision State-Based Learning

Consider a state transition from current state $s = (\boldsymbol{n}, f)$ to the next state $s' = (\boldsymbol{n}', f')$. We know from Eq. (5) that the transition dynamics has two components. This decomposition facilitates the design of RL algorithms that can effectively handle partial knowledge of the environment.

Let $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$ be a virtual state immediately after taking an action, but before the impact of the stochastic arrivals and departures, which is referred to as the post-decision state. Here $\tilde{\boldsymbol{n}}$ is the corresponding allocation matrix. Given this virtual state $\tilde{s}$, we can rewrite Eq. (8) as

$$\mathbb{P}(\boldsymbol{n}'|s, \boldsymbol{a}) = \mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}}) \cdot \mathbb{P}(\tilde{\boldsymbol{n}}|s, \boldsymbol{a}).$$

This enables us to decompose the system dynamics in Eq. (5) into action-controlled (known), $\mathbb{P}^k(\cdot|\cdot,\cdot)$, and stochastic (unknown), $\mathbb{P}^u(\cdot|\cdot,\cdot)$, transitions.

$$\mathbb{P}(s'|s, \boldsymbol{a}) = \underbrace{\mathbb{P}(\tilde{\boldsymbol{n}}|s, \boldsymbol{a})}_{\substack{\text{action controlled} \\ \mathbb{P}^k(\tilde{s}|s,\boldsymbol{a})}} \cdot \underbrace{\mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}}) \cdot \mathbb{P}(f')}_{\substack{\text{exogenous} \\ \mathbb{P}^u(s'|\tilde{s},\boldsymbol{a})}}. \tag{10}$$

The purely action controlled evolution of the allocation matrix is deterministic, i.e., for $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$,

$$\mathbb{P}^k(\tilde{s}|s, \boldsymbol{a}) = \mathbb{P}(\tilde{\boldsymbol{n}}|s, \boldsymbol{a}) = 1. \tag{11}$$

The uncertainty in system dynamics arises primarily from the stochastic arrival and departure processes, enabling a structured yet flexible modeling approach for learning-based control. We can leverage these structural attributes to utilize the potential of post-decision state analysis to reduce complexity [24], [25]. In the next section, we present reinforcement learning algorithms based on PDS to efficiently learn the optimal policy under partially known system dynamics.

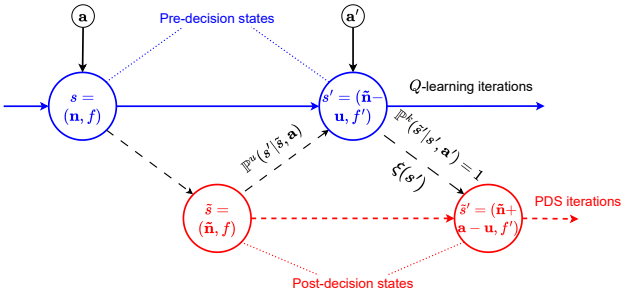


Fig. 3: Illustration of PDS-based state transition.

C. PDS Learning Algorithms: PDS Value Iteration (PDS-VI) and PDS-Q Learning

1) PDS-based value iteration (PDS-VI): Let $s = (\boldsymbol{n}, f)$ be the state of the system in some time slot. Let $a \in \mathcal{A}(s)$ be the action chosen at state s and $\boldsymbol{a}$ be the corresponding matrix. Then the post-decision state represented by $\tilde{s}$ is given by $\tilde{s} = (\tilde{\boldsymbol{n}}, f) = (\boldsymbol{n} + \boldsymbol{a}, f)$. Let $\boldsymbol{u}$ be the $K \times M$ matrix of departures observed at the end of the time slot whose row vectors, $\boldsymbol{u}_k$, $k \in [K]$, are given by

$$\boldsymbol{u}_k = \begin{cases} \boldsymbol{e}_m, & \text{flow of type } m \text{ departs from UPF } k \\ \boldsymbol{0}, & \text{if no departures.} \end{cases} \tag{12}$$

The next actual state, considering the stochastic arrivals and departures, is termed as the *pre-decision state*, $s' = (\boldsymbol{n}', f')$, where $\boldsymbol{n}' = \tilde{\boldsymbol{n}} - \boldsymbol{u} = \boldsymbol{n} + \boldsymbol{a} - \boldsymbol{u}$. The sequence of state transitions considering PDS is shown in Figure 3.

The Bellman equation for our MDP can be written as,

$$V(s) = \min_{a \in \mathcal{A}(s)} \left\{ \xi(s) + \gamma \sum_{s'} \mathbb{P}(s'|s, \boldsymbol{a}) \cdot V(s') \right\}. \tag{13}$$

By leveraging the PDS transition structure in Eqs. (10), (11) and substituting for s', we can rewrite Eq. (13) as

$$V(s) = \min_{a \in \mathcal{A}(s)} \left\{ \xi(s) + \gamma \sum_{f', \boldsymbol{u}} \mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}}) \cdot \mathbb{P}(f') \cdot V(\tilde{\boldsymbol{n}} - \boldsymbol{u}, f') \right\}.$$

Now we define the post-decision state value function, $\widetilde{V} : \mathcal{S} \to \mathbb{R}$ as the expected value over all pre-decision states, s', reachable from the post-decision state,

$$\widetilde{V}(\tilde{s}) = \mathbb{E}_{s'}[V(s')] = \sum_{s'} \mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}}) \cdot \mathbb{P}(f') \cdot V(s'). \tag{14}$$

With this definition, the Bellman equation becomes

$$V(s) = \min_{a \in \mathcal{A}(s)} \left\{ \xi(s) + \gamma \widetilde{V}(\tilde{s}) \right\}. \tag{15}$$

Let $a' \in \mathcal{A}(s')$ be the action chosen at state s' and $\boldsymbol{a}'$ be the corresponding indicator matrix. Also, let the next post decision state after selecting a' at s' be $\tilde{s}' = (\boldsymbol{n}' + \boldsymbol{a}', f')$. We can write the value equation for the state, s', as

$$V(s') = \min_{a' \in \mathcal{A}(s')} \left\{ \xi(\boldsymbol{n}', f') + \gamma \widetilde{V}(\boldsymbol{n}' + \boldsymbol{a}', f') \right\}.$$

Algorithm 1 Value iteration with PDS (PDS-VI)

Input: Number of UPFs K, Number of flow types M, Resource requirements $\bar{R}_m$, Capacities C_k, Unit cost c_k
Output: Policy π

Initialize post-state $\tilde{s} = (\tilde{\boldsymbol{n}}, f), \tilde{s} \in \mathcal{S}$, $\widetilde{V} \leftarrow \boldsymbol{0}$, and $t = 0$
for $t = 0, 1, 2, 3, \ldots$ **do**
  Observe departures and compute departure matrix $\boldsymbol{u}$ using Eq. (12) //departures
  Observe the flow arrival f' // arrival
  Compute $s' = (\tilde{\boldsymbol{n}} - \boldsymbol{u}, f')$
  Determine feasible actions, $\mathcal{A}(s')$
  Update $\widetilde{V}_{t+1}(\tilde{s})$ using Eq. (17) and obtain $\boldsymbol{a}'$ corresponding to the minimizing action $a' \in \mathcal{A}(s')$
  Monotonicity-based Acceleration:
  Find all immediate neighbor states of $\tilde{s}$ that differ by 1 flow in the allocation matrix
  if neighbor state matrix $\tilde{\boldsymbol{n}}''$ has 1 additional flow and $\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}'') < \widetilde{V}_{t+1}(\tilde{\boldsymbol{n}})$ **then** $\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}'') \leftarrow \widetilde{V}_{t+1}(\tilde{\boldsymbol{n}})$
  end if
  if neighbor state $\tilde{\boldsymbol{n}}''$ has 1 less flow and $\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}'') > \widetilde{V}_{t+1}(\tilde{\boldsymbol{n}})$ **then** $\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}'') \leftarrow \widetilde{V}_{t+1}(\tilde{\boldsymbol{n}})$
  end if
  Compute $\tilde{s}' = (\tilde{\boldsymbol{n}} - \boldsymbol{u} + \boldsymbol{a}', f')$
  Update $\pi(s') = a'$ and $\tilde{s} = \tilde{s}'$
end for
return π

substituting this in Eq. (14), we obtain the equation that forms the basis for the proposed value iteration algorithm,

$$\widetilde{V}(\tilde{s}) = \sum_{f', \boldsymbol{u}} \mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}}) \cdot \mathbb{P}(f') \min_{a' \in \mathcal{A}(s')} \Big[\xi(\tilde{\boldsymbol{n}} - \boldsymbol{u}, f') + \gamma \widetilde{V}(\tilde{\boldsymbol{n}} - \boldsymbol{u} + \boldsymbol{a}', f') \Big]. \tag{16}$$

Remark 1. *Compared to the standard Bellman equation, using the structural property of the formulation, the expectation is outside the minimization in Eq.* (16)*. This enables us to propose a value learning using stochastic approximation.*

Remark 2. *Eq.* (16) *is the Bellman's equation for the PDS value function $\widetilde{V}$. By substituting the optimal value corresponding to the PDS analysis, $\widetilde{V}^\star$, in Eq.*(15)*, the optimal value vector of Eq.* (13)*, $V^\star$, can be obtained. This in turn, also gives us the optimal policy $\pi^\star$.*

2) PDS-VI Algorithm and Convergence Guarantee: Next, we describe our proposed PDS-based value iteration (PDS-VI) algorithm. Consider Eq. (16). Let $\{\alpha_t\}_{t \geqslant 0}$ be a positive step-size sequence satisfying the Robbins-Monro conditions $\sum_t \alpha_t = \infty$ and $\sum_t (\alpha_t)^2 < \infty$. Using these step sizes, we perform stochastic approximation of value vector in Eq. (16) and update the estimate of each state at every time step depending on the observed arrivals and departures according

to the following update rule

$$\widetilde{V}_{t+1}(\tilde{s}) = \widetilde{V}_t(\tilde{s}) + \alpha_t \Bigg[\min_{a' \in \mathcal{A}(s')} \Big(\xi(\tilde{\boldsymbol{n}} - \boldsymbol{u}, f') + \gamma \widetilde{V}_t(\tilde{\boldsymbol{n}} - \boldsymbol{u} + \boldsymbol{a}', f') \Big) - \widetilde{V}_t(\tilde{s}) \Bigg], \tag{17}$$

$$\widetilde{V}_{t+1}(\tilde{s}'') = \widetilde{V}_t(\tilde{s}''), \qquad \text{for all } \tilde{s}'' \neq \tilde{s}.$$

The iterative value estimation algorithm based on the given update is provided in Algorithm 1. Theorem below proves that these iterates converge to the optimal value of the PDS value vector, $\widetilde{V}^\star$. Proof of Theorem 1 is given in Section IX.

Theorem 1. *The PDS value function iterates in Eq.* (17) *converge to the optimal PDS value function,* $\widetilde{V}_t \to \widetilde{V}^\star$. □

3) Properties of PDS-VI: In this subsection, we establish three key structural properties of the proposed PDS-VI approach and demonstrate how these properties can be leveraged to accelerate learning and improve convergence of the learning algorithms. These findings are further corroborated through simulation experiments. We provide the formal statements in Theorem 2 and present the proof in Section VIII.

Theorem 2. *The proposed PDS value iteration approach satisfies the following three structural properties.*

1) *For any post-decision state* $(\tilde{\boldsymbol{n}}, f)$*, the post-decision value function satisfies* $\widetilde{V}(\tilde{\boldsymbol{n}}, f) = \widetilde{V}(\tilde{\boldsymbol{n}}, \cdot)$*. That is, the PDS value function does not depend on flow arrival index* f*.*
2) *Consider two arbitrary post-decision states* $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$ *and* $\tilde{s}' = (\tilde{\boldsymbol{n}}', f)$ *such that* $\tilde{\boldsymbol{n}}' \geqslant \tilde{\boldsymbol{n}}$ *(element-wise). Then, the optimal post-decision value function* $\widetilde{V}^\star(\tilde{\boldsymbol{n}})$ *is monotonically non-decreasing, i.e.,* $\widetilde{V}^\star(\tilde{\boldsymbol{n}}') \geqslant \widetilde{V}^\star(\tilde{\boldsymbol{n}})$*.*
3) *Consider two arbitrary pre-decision states* $s = (\boldsymbol{n}, f)$ *and* $s' = (\boldsymbol{n}', f)$ *such that* $\boldsymbol{n}' \geqslant \boldsymbol{n}$ *(element-wise). For a given flow type arrival* f*, the optimal pre-decision value function is monotonically non-decreasing, i.e.,* $V^\star(\boldsymbol{n}, f) \leqslant V^\star(\boldsymbol{n}', f)$*, for all* $f \in \{0\} \cup [M]$*.* □

We utilize these properties to further accelerate learning. Specifically, for states with poor value estimate, we clip their value estimates to the values provided by their neighbors by leveraging the monotonicity property.

4) PDS Q-Learning and Convergence Guarantees: We now present a PDS-based Q learning (PDS-Q) algorithm. While PDS-VI updates the value estimate of a single state in each time slot, the PDS-Q algorithm allows multiple states to be updated within a single slot, potentially leading to faster convergence. PDS-Q has two key distinctions over the single update PDS-VI. (i) It enables action exploration which might aid in learning a better policy, and (ii) A given post-decision state can be reached from multiple pre-decision states under different actions, arrivals and departures. It is thus possible to update multiple pre-decision states from a single post-decision state value estimate using a two-stage PDS-Q algorithm.

Consider the transitions shown in Fig. 3. Let $\widetilde{Q} \in \mathcal{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ and $Q \in \mathcal{R}^{|\mathcal{S}| \times |\mathcal{A}|}$ be the Q-tables of the PDS state-action pairs $(\tilde{s}, \boldsymbol{a})$ and actual state-action pairs $(s, \boldsymbol{a})$, respectively. We know that $\widetilde{Q}(s, \boldsymbol{a})$ is the expected value of all possible future states that can be reached from state s under action $\boldsymbol{a}$. Then the Q-value of the PDS is given by

$$\begin{aligned} \widetilde{Q}(\tilde{s}, \boldsymbol{a}) &= \mathbb{E}_{\boldsymbol{s}'}[\gamma \min_{a'} Q(s', \boldsymbol{a}')] \\ &= \sum_{s'} \mathbb{P}^u(s' | \tilde{s}, \boldsymbol{a}) [\gamma \min_{\boldsymbol{a}'} Q(s', \boldsymbol{a}')] \\ &= \sum_{s'} \mathbb{P}(\boldsymbol{n}' | \tilde{\boldsymbol{n}}) \cdot \mathbb{P}(f') [\gamma \min_{a'} Q(s', \boldsymbol{a}')]. \end{aligned}$$

Then we can update $\widetilde{Q}$ for some state-action pair $(\tilde{s}, \boldsymbol{a})$ at some time slot $t+1$ using stochastic approximation,

$$\widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a}) = \begin{cases} \widetilde{Q}_t(\tilde{s}, \boldsymbol{a}) + \alpha_t \Big[\gamma \min_{\boldsymbol{a}'} Q_t(s', \boldsymbol{a}') \\ \quad - \widetilde{Q}_t(\tilde{s}, \boldsymbol{a}) \Big], & \text{if } (\tilde{s}, \boldsymbol{a}) = (\tilde{s}_t, \boldsymbol{a}_t); \\ \widetilde{Q}_t(\tilde{s}, \boldsymbol{a}), & \text{otherwise.} \end{cases} \tag{18}$$

This is then used to update the Q-value of the actual pre-decision state using the following equation:

$$\begin{aligned} Q_{t+1}(s, \boldsymbol{a}) &= \mathbb{E}[\xi(s) + \widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a})] \\ &= \xi(s) + \mathbb{P}^k(\tilde{s} | s, \boldsymbol{a}) \widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a}) \\ &= \xi(s) + \widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a}). \end{aligned} \tag{19}$$

Theorem 3 proves convergence of PDS-Q algorithm. The proof is given in Section IX.

Theorem 3. *The iterates of the two-stage PDS-Q algorithm in Eqs.* (18) *and* (19) *converge to the optimal value functions* $(Q_t \to Q^\star)$ *and* $(\widetilde{Q}_t \to \widetilde{Q}^\star)$ *respectively.* □

Remark 3. *PDS-Q adds significant advantage in systems with large or highly stochastic action spaces, where multiple pre-decision states can lead to the same post-decision state under different actions. In such cases, a single post-decision Q-value update can be used to update multiple pre-decision state Q-values simultaneously, enabling faster convergence.*

Algorithm 2 PDS Q-Learning (PDS-Q)

Input: Number of UPFs K, Number of flow types M, Resource requirements $\bar{R}_m$, Capacities C_k, and Unit cost c_k
Output: Policy π

Initialize $\widetilde{Q} \leftarrow 0$, $Q \leftarrow 0$, $s = (\boldsymbol{n}, f), s \in \mathcal{S}$, and $t = 0$
for $t = 0, 1, 2, 3, \ldots$ **do**
  Determine feasible actions, $\mathcal{A}(s)$
  Take an action $a \in \begin{cases} \arg\min_{a \in \mathcal{A}} Q(s, \boldsymbol{a}), & \text{w.p. } 1 - \epsilon \\ \text{random action}, & \text{w.p. } \epsilon. \end{cases}$
  Find the corresponding matrix $\boldsymbol{a}$
  Compute $\tilde{\boldsymbol{n}} = \boldsymbol{n} + \boldsymbol{a}$ and $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$
  Observe departures and compute departure matrix $\boldsymbol{u}$ using Eq. (12) //departures
  Observe the flow arrival f' // arrival
  Compute $s' = (\tilde{\boldsymbol{n}} - \boldsymbol{u}, f')$
  Update $\widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a})$ using Eq. (18)
  Update $Q_{t+1}(s, \boldsymbol{a})$ using Eq. (19)
  Update $\pi(s) = a$ and $s = s'$
end for
return π

Algorithm 3 Accelerated PDS-based Tournament Algorithm

Input: Number of UPFs K, Number of flow types M, Resource requirements $\bar{R}_m$, Capacities C_k, and Unit cost c_k
Output: Allocation policy π_T,

1: **for** each UPF pair $i, j \in [K]$ **do**
2: Run Algorithm 1 to solve the problem for UPFs i and j
3: Obtain optimal PDS policy for UPFs i and j, $\pi^\star_{ij}$
4: **end for**
5: **for** each state $s \in \mathcal{S}$ **do**
6: **for** $i, j \in [K]$, $i = 1$, $j = 2$ **do**
7: Find the sub-state of s restricted to UPFs i, j, $s|_{i,j}$
8: Find the allocation, $a \in \mathcal{A}(s|_{i,j})$ for $s|_{i,j}$ under $\pi^\star_{ij}$
9: **if** $a = i$ **then** $i = i$ and $j = j + 1$
10: **else** $i = j$ and $j = j + 1$
11: **end if**
12: Go to line 7
13: **end for**
14: $\pi_T(s) = i$
15: **end for**
16: **return** π_T

VI. Approximate Tournament PDS Algorithm

Reinforcement learning algorithms often suffer from the curse of dimensionality, as the state space grows exponentially in the number of UPFs. However, considering the structural properties of the state space in our formulation, we observe that given a state matrix $\boldsymbol{n}$, the rows are independent, i.e., the number of flows in each UPF and their transition dynamics depend solely on the chosen action and exogenous factors and are independent of those of the other UPFs. We exploit this property to decompose the original problem into smaller sub-problems and develop a novel accelerated heuristic algorithm that solves a set of smaller sub-problems involving fewer candidate UPFs rather than optimizing over all UPFs at once and then combine the policy to derive a strategy for the original problem. Our proposition is that a complex K-UPF decision problem can be decomposed as a series of 2-UPF decisions and the sub-policies can be aggregated to obtain an approximate global policy π_T for the K-UPF problem. Specifically, we compute the optimal policies $\pi^\star_{ij}$ for all $\binom{K}{2}$ pairs of UPFs using Algorithm 1, where (i, j) are the considered UPF pair. For each state in the original problem, we obtain the policy by comparing and aggregating the pairwise policies.

Definition 1. *Consider the state $s = (\boldsymbol{n}, f)$. For a given pair of UPFs i, j, we define the sub-state as $s_{ij} = (\boldsymbol{n}_{ij}, f)$ where $\boldsymbol{n}_{ij}$ is the $2 \times M$ sub-matrix of $\boldsymbol{n}$ containing rows $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$. The arriving flow f is preserved.*

To obtain an approximate policy over K UPFs, we use a sequential elimination tournament. We begin with a pair of UPFs and compute the sub-state s_{ij}. From the corresponding pairwise optimal policy $\pi^\star_{ij}$, we extract the optimal action for s_{ij} and retain the UPF associated with the chosen action, while discarding the other. The retained UPF is then compared with the next candidate UPF, and the process is repeated until all K UPFs have been evaluated and we obtain an approximate action (UPF), which amounts to $K-1$ comparisons. If at any step, both UPFs in the pair are infeasible for the arriving flow, we continue with the previously retained UPF. This choice does not affect the final outcome, since any feasible UPF compared later will prevail, and if none exists then the optimal action is 0 (i.e., the flow is blocked). We repeat this process for each state s to aggregate the heuristic action $\pi_T(s)$ thus constructing the heuristic policy π_T for the original problem. This heuristic method is provided in Algorithm 3.

In practice, the algorithm requires only $K-1$ lookups for a dynamic K-UPF decision as the optimal policies for all $\binom{K}{2}$ sub-problems can be precomputed offline. This heuristic policy does not guarantee optimality, but it drastically reduces the computational time by lowering the per-decision computation from exponential in K to linear in K while providing an approximate optimal solution.

Remark 4. *The proposed Tournament algorithm can be easily adapted to scenarios where new UPFs are added to the system. When a new UPF is added to the system, only K additional pairwise policies between the new UPF and each existing one needs to be computed. Consequently, the online decision process involves one additional comparison, maintaining linear computational growth in K ensuring the scalability of the Tournament policy.*

Complexity Analysis: A summary of the storage and computational complexities for each algorithm is provided in Table II. In terms of storage, The value iteration algorithm needs to store the value estimate at each state and hence has a complexity of $|S|$. Both Q-learning algorithms stores the expected value of each state-action pair and thus the complexity of $|\mathcal{S}| \times |\mathcal{A}|$. When considering computational complexity, each algorithm performs a minimization over the feasible actions for the expected value so each has the complexity of $|\mathcal{A}|$. For the Tournament method, each pairwise policy needs to be stored leading to a storage complexity of $\binom{K}{2} \approx K^2\times$ the state space cardinality of the two UPF subproblem. The computational complexity is $O(|A|)$ for $K-1$ lookups.

TABLE II: Complexity Analysis

Algorithm	Storage Complexity	Computational Complexity
Q-learning	$\mathcal{O}(\|\mathcal{S}\| \times \|\mathcal{A}\|)$	$\mathcal{O}(\|\mathcal{A}\|)$
Q-learning with PDS	$\mathcal{O}(\|\mathcal{S}\| \times \|\mathcal{A}\|)$	$\mathcal{O}(\|\mathcal{A}\|)$
Value iteration with PDS	$\mathcal{O}(\|\mathcal{S}\|)$	$\mathcal{O}(\|\mathcal{A}\|)$
Tournament method	$O(\|\mathcal{A}\|^2 \cdot (\lfloor C/\bar{R}_1 \rfloor)^{2M})$	$\mathcal{O}(\|\mathcal{A}\|)$

VII. Numerical Simulations

A. Data Generation and Experimental Setting

We evaluated the proposed algorithms on a synthetic dataset. The following parameters are fixed in all experiments. In all experiments, we considered two flow types w.p. $b_1 = 0.6$ and $b_2 = 0.4$, respectively. The average rate requirement for each flow type is set as $\bar{R}_1 = 30$ and $\bar{R}_2 = 35$. The maximum data rate C_k is set to 100 for each UPF. We set the discount factor to 0.96. We conducted multiple experiments by varying

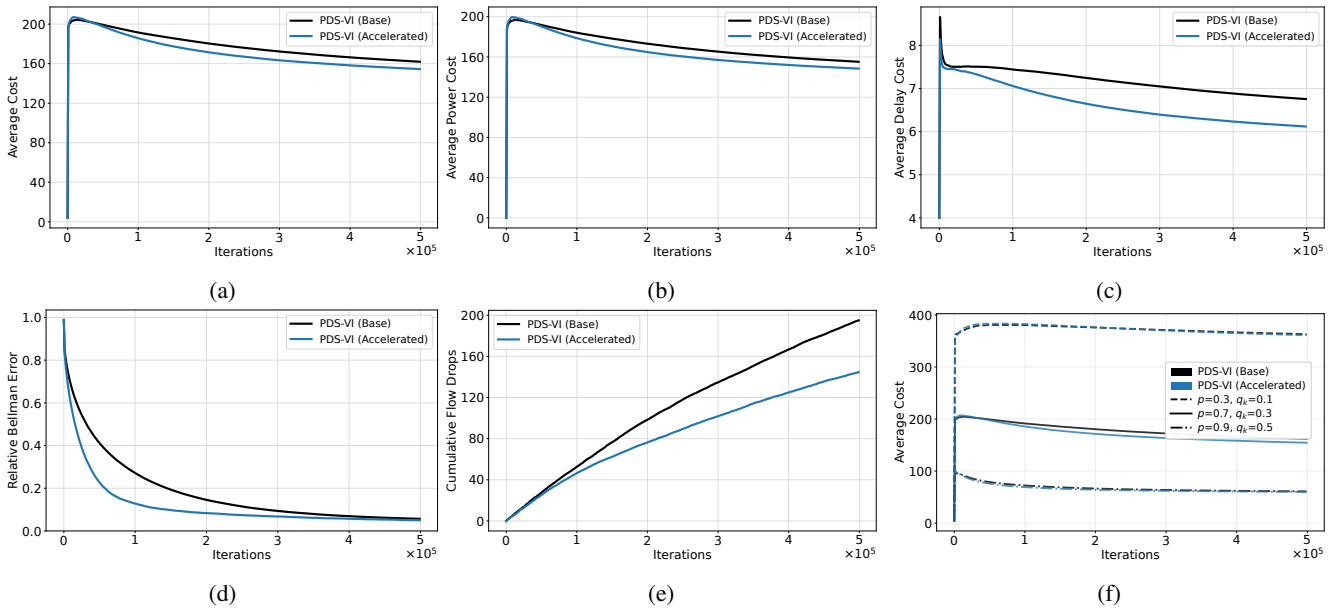


Fig. 4: Comparison of PDS-VI (Base) vs. PDS-VI (Accelerated) with monotonicity. For all plots the number of UPFs $K = 4$ and the number of flow types $M = 2$. Figure 4a presents average cost, Figure 4b presents average power cost, Figure 4c presents average delay cost, Figure 4d presents plots for average relative Bellman error, Figure 4e represents the number of flows blocked with respect to iteration, and Figure 4f presents an ablation study for the 4 UPF case.

the number of UPFs, as well as the arrival and departure probabilities, and compared the performance against various baselines. We implemented all simulations in Python. The simulations were run on a Windows machine with *Intel(R) Xeon(R) W-1370* processor. All simulation results are averaged over 100 independent Monte Carlo runs.

B. Baseline Algorithms

For experiments with smaller problem sizes, where computing the optimal policy is feasible, we derived the ground-truth optimal policy using value iteration. Note that this requires knowledge of the exogenous parameters. The resulting optimal policy is then used to compute error metrics for evaluating the performance of the proposed learning algorithm relative to other learning-based baselines.

Value iteration is a dynamic programming algorithm that aims to find the optimal value function $V^\star(s)$ that gives the expected minimum cost starting from state s and following the optimal policy thereafter. It can be viewed as an update rule version of the Bellman optimality equation. The update equation for value iteration is given by

$$V_{t+1}(s) = \xi(s) + \min_{a \in \mathcal{A}(s)} \gamma \sum_{s'} \mathbb{P}(s'|s, \boldsymbol{a}) V_t(s') \quad \forall s \in \mathcal{S}. \quad (20)$$

We also considered the standard Q-learning algorithm introduced in Eq. (9) as a baseline to compare the performance of the proposed algorithms. The value function $V(s) = \min_a Q[s, a]$. The Q-learning algorithm is implemented with ϵ-greedy exploration with $\epsilon = 0.3$. The Q-learning algorithm does not exploit the underlying structure of the problem. Our objective is to demonstrate that the proposed PDS approach achieves performance comparable to standard Q-learning more efficiently, i.e., with fewer data samples and reduced computational time. For a scalable baseline for larger sized problems, we utilize a greedy heuristic that chooses the action with minimum immediate cost as a baseline method.

C. Metrics

We compared the algorithms in terms of convergence speed and average cost. To this end, we consider the time-averaged cost incurred by each algorithm, given by

$$\bar{\xi}_t = \frac{1}{t} \sum_{i=1}^{t} \xi(s_i),$$

for cost performance and the convergence by relative Bellman error (RBE). Since states are visited at different frequencies, we opt for a weighted error, given by

$$RBE_t = \frac{\sum_s w_s \cdot (|V_t(s) - V^\star(s)|)}{\sum_s w_s \cdot (|V^\star(s)|)},$$

where w_s is number of times the state s is visited and $V^\star$ is the ground truth. We initialize the value vector and Q-table with the immediate cost to provide a fair starting point.

D. Results and Discussion

1) Experiment 1 - Monotonicity Analysis: In this experiment we studied the impact of leveraging the monotonicity property of the post-decision state value vector $\widetilde{V}$. We set the parameters as $K = 4$, $p = 0.7$, and $q_k = 0.3$, for all $k \in [K]$. The total number of states in this experiment is 12288. The individual power costs are $c_1 = 4, c_2 = 3, c_3 = 2$, and $c_4 = 1$. For the ablation study, we consider two additional flow arrival and departure probabilities with $p = 0.3, q_k = 0.1$ and $p = 0.9, q_k = 0.5$. The results are presented in Figure 4.

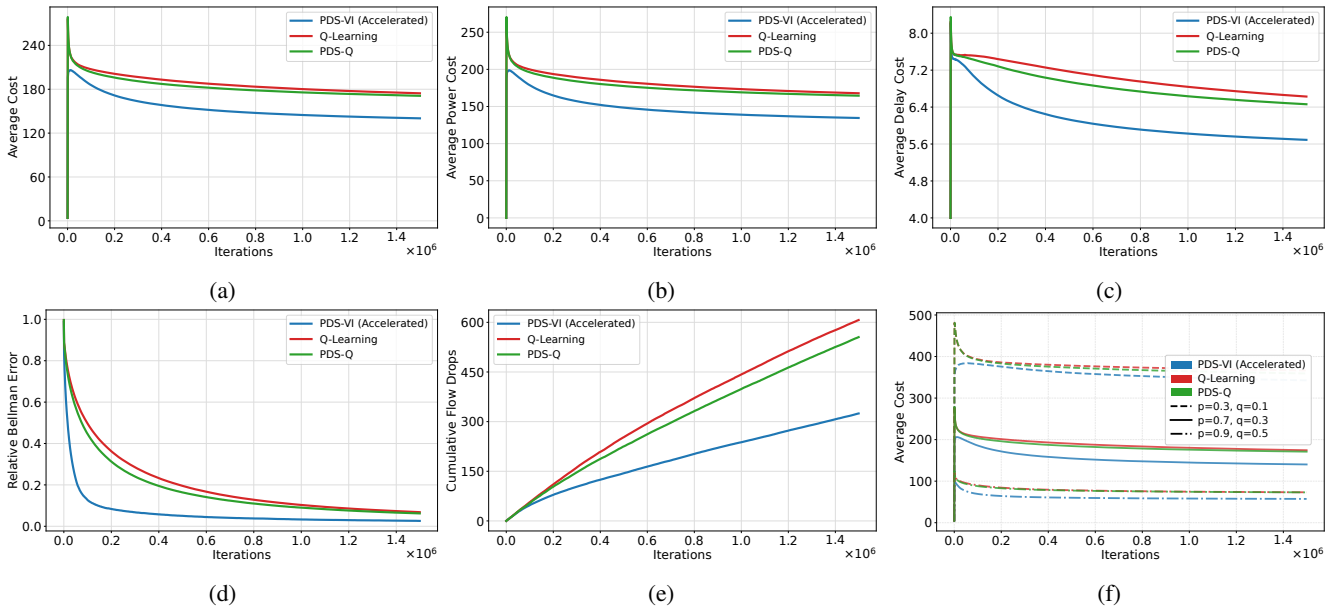


Fig. 5: Comparison of PDS algorithms vs. Q-learning. For all plots the number of UPFs $K = 4$ and the number of flow types $M = 2$. Figure 5a presents average cost, Figure 5b presents average power cost, Figure 5c presents average delay cost, Figure 5d presents plots for average relative Bellman error, RBE, Figure 5e represents the number of flows dropped with respect to iteration, and Figure 5f presents the results for an ablation study by varying the exogeneous parameters.

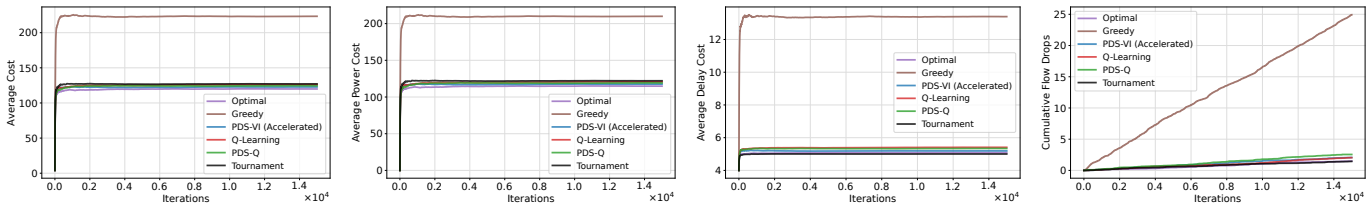
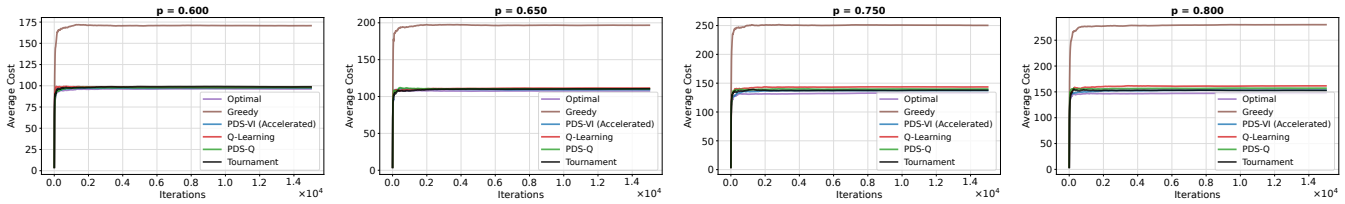


Fig. 6: Evaluation of learned policies - Optimal (Dynamic Programming), Greedy, PDS-VI (Accelerated), Q-PDS, Q-learning, and Tournament algorithms. For all plots $K = 4$ and $M = 2$. Figure 6a, Figure 6b, Figure 6c presents the average cost, average power cost, and average delay cost, respectively. Figure 6d represents the number of flows dropped with respect to iteration.

The PDS-VI (Accelerated) exploiting the monotonicity property achieves faster convergence in terms of relative Bellman error (RBE), reaching RBE < 0.1 at approximately 2×10^5 iterations compared to PDS-VI (Base) at 3×10^5 iterations. PDS-VI (Accelerated) also delivers improvements in other metrics. It reduces cumulative flow rejections by approximately 50 flows, achieves lower delay cost (6.2 vs. 6.8), and maintains lower total and power costs than PDS-VI (Base). These results demonstrate that exploiting monotonicity properties yields tangible benefits in network performance, particularly, faster convergence, flow acceptance, and delay efficiency. We conducted an ablation study by varying the arrival and departure probabilities to assess the sensitivity of the learning algorithms. Figure 4f highlights a key structural effect. When the departure probabilities are small, the system explores the state space sufficiently, leaving no margin for additional gains from monotonicity-based bounds. Conversely, when the departure probabilities are sufficiently high, the trajectory visits too few distinct states, so the algorithm cannot effectively exploit monotonicity-based acceleration.

2) Experiment 2 - Training Phase for Learning Algorithms: This experiment aims to evaluate the performance of the learning algorithm during the training phase. We set the baseline parameters as $K = 4$, $p = 0.7$, and $q_k = 0.3$, for all $k \in [K]$. The total number of states for this example is 12288. The individual power costs are $c_1 = 4, c_2 = 3, c_3 = 2$, and $c_4 = 1$. For the ablation study, we considered two additional flow arrival and departure probabilities with $p = 0.3, q_k = 0.1$ and $p = 0.9, q_k = 0.5$. The results are presented in Figure 5.

As shown in Figure 5d, the PDS-based algorithms converge faster than the standard Q-learning algorithm. While the PDS-VI (Accelerated) achieves an RBE < 0.1 in just 2×10^5 iterations, Q-learning with PDS achieves it in under 1×10^6 iterations compared to standard Q-learning which requires 1.5×10^6 iterations. This rapid convergence in RBE directly translates to faster convergence of the underlying cost metrics. As evidenced by Figures 5a-5c, PDS-VI (Accelerated) achieves more efficient cost performance, reaching a lower average cost fastest. PDS-Q also performs better than baseline Q-learning although the difference is not as significant as PDS-



(a) (b) (c) (d)

Fig. 7: Robustness evaluation of learned policies under varied arrival probability. For all plots, $K = 4$ and $M = 2$. Figures 7a, 7b, 7c, and 7d compare cost performance across Optimal (Dynamic Programming with $p = 0.7$), Greedy, PDS-VI (Accelerated), PDS-Q, Q-learning, and Tournament algorithm baselines for arrival probability deviations of $p - 0.1$, $p - 0.05$, $p + 0.05$, and $p + 0.1$, respectively. Results demonstrate the stability of structure-aware methods under distribution shift.

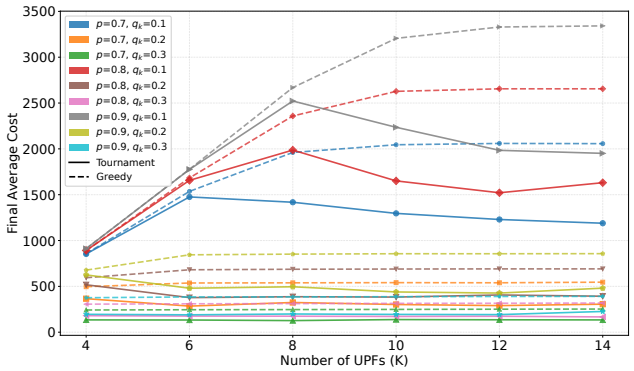


Fig. 8: Performance comparison of Greedy vs. Tournament policies across increasing number of UPFs K and different arrival (p) and departure (q_k) probabilities.

VI (Accelerated). This can be attributed to two main factors: (i) The Q-algorithms perform exploration during the learning phase, leading to sub-optimal actions and higher costs, and (ii) since the state transition (the allocation matrix of the state) is deterministic for a given action, this implies that action selection is not benefiting from exploration and thus value iteration, which always chooses $\min_{\boldsymbol{a}}$, learns a policy similar to Q-based algorithms.

Figures 5b and 5c highlight the individual power and delay components which clearly emphasise that each component cost also mirrors similar effect: PDS-VI (Accelerated) achieves an average power cost of ≈ 130, improving over PDS-Q (≈ 170) and Q-learning (≈ 175), and maintains the smallest delay cost (≈ 5.7) relative to PDS-Q (≈ 6.5) and Q-learning (≈ 6.7). Figure 5e also confirms that PDS-VI (Accelerated), which chooses the minimizing action admits more flows, blocking fewer flows than PDS-Q and Q-learning. The ablation study demonstrates that, irrespective of the arrival and departure dynamics, PDS-based algorithms consistently outperform the standard Q-learning algorithm.

3) Experiment 3 - Policy Evaluation: In this experiment we evaluated the performance of policies obtained using Optimal Baseline (dynamic programming via value iteration), PDS-VI (Accelerated), PDS-Q, Q-learning, Tournament, and Greedy heuristic. We set the parameters as $K = 4$, $p = 0.7$, and $q_k = 0.3$, for all $k \in [K]$, and $|\mathcal{S}| = 12288$. The power costs are $c_1 = 4, c_2 = 3, c_3 = 2,$ and $c_4 = 1$. The results are presented in Figure 6.

During the evaluation of learned policies, the proposed methods - PDS-Q, PDS-VI (Accelerated), and Tournament algorithms perform comparably as seen in Figures 6a–6d. The PDS-VI (Accelerated), PDS-Q and Tournament algorithms achieve average cost close to the optimal baseline and better than Q-learning. The delay and power costs also show similar trends. Figure 6d highlights a crucial inference. Across 10^4 iterations with 7×10^3 arriving flows, all learning algorithms and the Tournament algorithm drop very few flows ($\approx$ 2–3), whereas the Greedy heuristic drops $\approx$ 25 flows. Greedy heuristic drops $\approx$ 22 flows more than the proposed approaches. However, while dropping fewer flows, the proposed approaches also achieve a much better cost performance. This is because the cost of serving ≈ 22 additional flows is much lesser compared to the substantial costs incurred by Greedy heuristic due to suboptimal decisions.

4) Experiment 4 - Robustness: In this experiment we studied the robustness of the policies trained at $p = 0.7$ when deployed under shifted arrival probability distributions. We set the parameters as $K = 4$ and $q_k = 0.3$, for all $k \in [K]$. The total number of states for this example is 12288. The individual power costs are $c_1 = 4, c_2 = 3, c_3 = 2,$ and $c_4 = 1$. Policies are evaluated under $p \in \{0.6, 0.65, 0.75, 0.8\}$, corresponding to ± 0.05 and ± 0.1 deviations from the training distribution ($p = 0.7$). We present the results in Figure 7.

Figure 7a ($p = 0.6$), Figure 7b ($p = 0.65$), Figure 7c ($p = 0.75$), and Figure 7d ($p = 0.8$) show average cost under each shifted distribution. Here, 'Optimal' refers to the policy obtained by dynamic programming solution at $p = 0.7$ which provides the baseline for variations in cost as arrival probability deviates from the training value, serving as a lower bound reference rather than the true optimum for each shifted distribution. Despite this distribution mismatch, PDS-VI (Accelerated) maintains performance closest to this reference across all arrival probability shifts. Q-learning exhibits greater degradation under higher arrival rates ($p = 0.8$), while Greedy consistently underperforms. The Tournament heuristic achieves robustness comparable to learning-based methods and for higher p even outperforms Q-PDS. These results demonstrate that policies learned by exploiting problem structure exhibit superior robustness to distribution shifts.

5) Experiment 5 - Scalability Analysis: This experiment evaluates the scalability of the Tournament method and as-

sesses how increasing the number of UPFs, K, affects cost, in comparison with a Greedy heuristic. We considered different configurations of (p, q_k) with $p \in \{0.7, 0.8, 0.9\}$ and $q_k \in \{0.1, 0.2, 0.3\}$ set uniformly across all $k \in [K]$. The number of UPFs K is varied from 4 to 14 in increments of 2. The c_k values are varied accordingly to ensure they are distinct for each $k \in [K]$. The results are presented in Figure 8.

Across the (p, q_k) configurations, Tournament outperforms Greedy for all values of K. The gap is largest when arrivals are higher and departures are lower (high p, low q_k), because congestion is higher and allocation decisions matter more. When departures are higher (larger q_k), congestion is lower, and the gap shrinks because the system is more forgiving to suboptimal decisions. Moreover, for each fixed (p, q_k) configuration, increasing the number of UPFs K is beneficial only up to a certain point. Beyond a certain threshold, adding UPFs does not significantly reduce the average cost and can be turned off to save energy.

VIII. Proof Of Theorem 2 (Structural Properties)

In this section we provide the proof of the properties of the PDS-VI analysis given in Theorem 2.

A. Preliminaries

Here we provide the background and standard results utilized in the proof of Theorem 2.

Definition 2 ([26]). *Let X and Y be two random vectors with distributions Δ and Δ' respectively. A coupling of random vectors X and Y is a pair $(\hat{X}, \hat{Y})$ defined on a common probability space such that the marginal distribution of $\hat{X}$ is Δ and the marginal distribution of $\hat{Y}$ is Δ'. The joint distribution of $(\hat{X}, \hat{Y})$ can be arbitrary, provided the marginals match.*

We first state a standard result connecting coupling to the ordering of expectations we use in the proof.

Proposition 1 ([26], [27]). ***(Monotone Coupling of Stochastic Dominance)*** *Let X and Y be two random vectors. If there exists a coupling $(\hat{X}, \hat{Y})$ of X and Y such that $\hat{X} \leqslant \hat{Y}$ almost surely (element-wise), then X is stochastically dominated by Y. Specifically, for any component-wise non-decreasing function g, provided that the expectations are well-defined,*

$$\mathbb{E}[g(X)] \leqslant \mathbb{E}[g(Y)].$$

B. Supporting Lemmas and their Proofs

In this subsection, we present the supporting lemmas used in the proof of Theorem 2.

Consider two arbitrary post-decision states $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$ and $\tilde{s}' = (\tilde{\boldsymbol{n}}', f)$ that are identical in all components except that $\tilde{\boldsymbol{n}}'$ differs from $\tilde{\boldsymbol{n}}$ by one additional flow of type m' allocated to UPF k. That is, $\tilde{\boldsymbol{n}}'_{km'} = \tilde{\boldsymbol{n}}_{km'} + 1$ and $\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km} = \sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km} + 1$. We focus on the departure probabilities from UPF k, since all other UPFs are identical in both states. We have the following result on UPF k.

Lemma 1. *Consider two arbitrary post-decision states $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$ and $\tilde{s}' = (\tilde{\boldsymbol{n}}', f)$ such that they are identical except that $\tilde{\boldsymbol{n}}'$ differs from $\tilde{\boldsymbol{n}}$ by one additional flow of type m' allocated to UPF k. Assume a flow departs from the states $\tilde{s}$ and $\tilde{s}'$. Let q_{km} and q'_{km} be the probabilities that a flow of type m departs from the state $\tilde{s}$ and $\tilde{s}'$, respectively. Then,*

$$\sum_{m\neq m'} \big(q_{km} - q'_{km}\big) \;=\; q'_{km'} - q_{km'}.$$

Proof. Recall the construction of the states $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$ and $\tilde{s}' = (\tilde{\boldsymbol{n}}', f)$. We know that the matrices $\tilde{\boldsymbol{n}}$ and $\tilde{\boldsymbol{n}}'$ are identical except that $\tilde{\boldsymbol{n}}'$ has one additional flow of type m' allocated to UPF k. Focus on row k. Using the conditional probabilities q_{km} and q'_{km} defined above, we observe that for $m \neq m'$

$$\begin{aligned} q_{km} - q'_{km} &= \frac{\tilde{\boldsymbol{n}}_{km}}{\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km}} - \frac{\tilde{\boldsymbol{n}}_{km}}{\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km}} \\ &= \frac{\tilde{\boldsymbol{n}}_{km}}{(\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km})(\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km})}. \end{aligned}$$

Summing over all $m \neq m'$

$$\begin{aligned} \sum_{m\neq m'} (q_{km} - q'_{km}) &= \frac{\sum_{m\neq m'} \tilde{\boldsymbol{n}}_{km}}{(\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km})(\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km})} \\ &= \frac{\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km} - \tilde{\boldsymbol{n}}_{km'}}{(\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km})(\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km})}. \end{aligned}$$

We have

$$\begin{aligned} q'_{km'} - q_{km'} &= \frac{\tilde{\boldsymbol{n}}'_{km'}}{\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km}} - \frac{\tilde{\boldsymbol{n}}_{km'}}{\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km}} \\ &= \frac{\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km} - \tilde{\boldsymbol{n}}_{km'}}{(\sum_{m\in[M]} \tilde{\boldsymbol{n}}_{km})(\sum_{m\in[M]} \tilde{\boldsymbol{n}}'_{km})}. \end{aligned}$$

The two expressions are equal, hence proved. □

To apply Proposition 1 and establish monotonicity of the value function, we demonstrate the existence of a coupling of the departing flow types that ensures an ordering between the post-departure allocation matrices. In the following lemma, we first establish a coupling conditioned on the departure event.

Lemma 2. *Assume a flow departs from the states $\tilde{s}$ and $\tilde{s}'$. Let U and U' be random variables representing the departing flow types from UPF k in states $\tilde{\boldsymbol{n}}$ and $\tilde{\boldsymbol{n}}'$, respectively. There exists a coupling $(\tilde{U}, \tilde{U}')$ such that*

1) $\mathbb{P}(\tilde{U} \neq \tilde{U}') = q'_{km'} - q_{km'}$,
2) *If $\tilde{U} \neq \tilde{U}'$, then $\tilde{U}' = m'$.*

This coupling maximizes agreement between U and U' while concentrating all disagreement at type m'.

Proof. Define the overlap masses

$$h_m = \min\{q_{km}, q'_{km}\}, \qquad H = \sum_{m\in[M]} h_m.$$

We construct $(\tilde{U}, \tilde{U}')$ in the common probability space as follows.

a) with probability H, set $\tilde{U} = \tilde{U}'$. The probability of $\tilde{U} = \tilde{U}' = m$ is h_m/H
b) with probability $1 - H$, sample $\tilde{U}$ from the normalized residual of q which is $\frac{q_{km}-h_m}{1-H}$ and $\tilde{U}'$ from the normalized residual of q' which is $\frac{q'_{km}-h_m}{1-H}$.

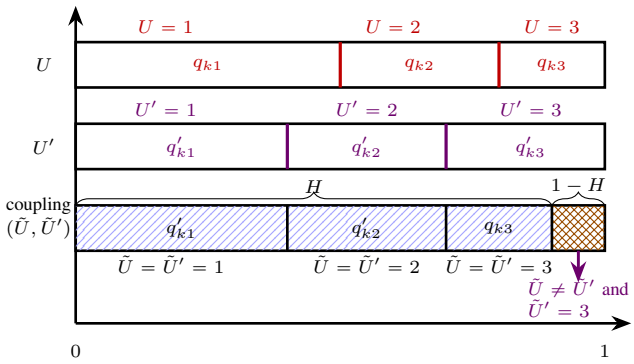


Fig. 9: Coupling construction example for Lemma 2 with $M = 3$ and $m' = 3$. The top two bars show the departure-type distributions q_{km} and q'_{km}, and the bottom bar shows a coupling $(\tilde{U}, \tilde{U}')$ constructed by partitioning $[0, 1]$ into $[0, H]$ and $(H, 1]$, where $h_m = \min\{q_{km}, q'_{km}\}$ and $H = \sum_m h_m$. On $[0, H]$, the coupling sets $\tilde{U} = \tilde{U}' = m$ with $\mathbb{P}(\tilde{U} = \tilde{U}' = m) = h_m$. On $(H, 1]$, the remaining mass is $1 - H = q'_{k3} - q_{k3}$ and yields $\tilde{U}' = 3$, so $\mathbb{P}(\tilde{U} \neq \tilde{U}') = q'_{k3} - q_{k3}$ and $\tilde{U}' = 3$ whenever $\tilde{U} \neq \tilde{U}'$.

By construction, the marginals match as shown

$$\mathbb{P}(\tilde{U} = m) = H \cdot \frac{h_m}{H} + (1 - H) \cdot \frac{q_{km} - h_m}{1 - H} = q_{km},$$
$$\mathbb{P}(\tilde{U}' = m) = H \cdot \frac{h_m}{H} + (1 - H) \cdot \frac{q'_{km} - h_m}{1 - H} = q'_{km}.$$

Thus $(\tilde{U}, \tilde{U}')$ is a coupling.

Now we prove 1) and 2) in the lemma statement hold for this coupling. By Lemma 1, $q_{km} \geqslant q'_{km}$ for all $m \neq m'$ and $q'_{km'} > q_{km'}$. Thus $h_m = \min\{q_{km}, q'_{km}\} = q'_{km}$ for $m \neq m'$.

For 1), $\tilde{U} \neq \tilde{U}'$, from b) $\tilde{U}'$ follows the normalized residual distribution $\frac{q'_{km} - h_m}{1 - H}$.

$$\begin{aligned}
\mathbb{P}(\tilde{U} \neq \tilde{U}') &= 1 - H = 1 - \sum_m \min\{q_{km}, q'_{km}\} \\
&= \sum_m (q_{km} - \min\{q_{km}, q'_{km}\}) \\
&= \sum_{m \neq m'} (q_{km} - q'_{km}) \quad \text{(residual zero at } m') \\
&= q'_{km'} - q_{km'} \quad \text{(by Lemma 1)}.
\end{aligned}$$

For 2), by Lemma 1, the residual $\frac{q'_{km} - h_m}{1 - H} = 0$ for $m \neq m'$ and positive only at m'. Therefore, $\tilde{U}' = m'$ almost surely in case b). This completes the proof. □

We provide an illustrative example to understand Lemma 2. Consider $M = 3$ and let $m' = 3$, so that $\tilde{\boldsymbol{n}}'$ differs from $\tilde{\boldsymbol{n}}$ by one additional flow of type 3 at UPF k, i.e., $\tilde{\boldsymbol{n}}'_{k3} = \tilde{\boldsymbol{n}}_{k3} + 1$. The coupling described in Lemma 2 is visualized in Fig. 9.

Define the overlap masses $h_m = \min\{q_{km}, q'_{km}\}$ and let $H = \sum_{m \in [M]} h_m$. To construct the coupling, draw a random seed $X \sim \mathrm{Unif}(0, 1)$. Partition $[0, 1]$ into $[0, H]$ and $(H, 1]$. If $X \leqslant H$, select a type $m \in \{1, 2, 3\}$ with probability h_m / H and set $\tilde{U} = \tilde{U}' = m$; equivalently, $\mathbb{P}(\tilde{U} = \tilde{U}' = m) = h_m$. If $X > H$, sample $\tilde{U}$ according to $\{(q_{km} - h_m)/(1 - H)\}_{m \in [M]}$ and sample $\tilde{U}'$ according to $\{(q'_{km} - h_m)/(1 - H)\}_{m \in [M]}$, which preserves the marginals.

Since $\tilde{\boldsymbol{n}}'$ has one extra flow of type 3, Lemma 1 implies that the imbalance between $\{q_{km}\}$ and $\{q'_{km}\}$ is concentrated at $m' = 3$. Hence $q'_{km} - h_m = 0$ for $m \neq 3$ and is positive only for $m = 3$. Therefore, on $\{X > H\}$ we have $\tilde{U}' = 3$ almost surely, and the mismatch probability is exactly the excess mass at flow type 3, i.e., $\mathbb{P}(\tilde{U} \neq \tilde{U}') = 1 - H = q'_{k3} - q_{k3}$.

Remark 5 extends the coupling in Lemma 2 to be unconditional on departures and ensures the stochastic dominance.

Remark 5. *Lemma 2 constructs a coupling of the departing flow types $(\tilde{U}, \tilde{U}')$, conditional on a departure from UPF k in both states. Since the departure probability of UPF k is q_k for both $\tilde{\boldsymbol{n}}_k$ and $\tilde{\boldsymbol{n}}'_k$, we can make the coupling unconditional on departures by coupling whether a departure occurs: with probability q_k a flow departs from both $\tilde{\boldsymbol{n}}_k$ and $\tilde{\boldsymbol{n}}'_k$ and with probability $1 - q_k$ no flow departs from either. If no departure occurs, set $\tilde{U} = \tilde{U}' = 0$; otherwise sample $(\tilde{U}, \tilde{U}')$ using the coupling in Lemma 2. We can construct such a coupling as the probability of departure is independent of the flow-type decision. Under this construction, either there are no departures from both states or in the event of a departure, we employ the coupling $(\tilde{U}, \tilde{U}')$ from Lemma 2 both of which preserves stochastic ordering.*

Lemma 3. *Consider the coupling $(\tilde{U}, \tilde{U}')$ constructed in the proof of Lemma 2. Then, under $(\tilde{U}, \tilde{U}')$, the following element-wise relation holds for the allocation matrices.*

$$\tilde{\boldsymbol{n}} - \boldsymbol{u} \leqslant \tilde{\boldsymbol{n}}' - \boldsymbol{u}'.$$

Proof. Recall the construction of the states $\tilde{s} = (\tilde{\boldsymbol{n}}, f)$ and $\tilde{s}' = (\tilde{\boldsymbol{n}}', f)$. The matrices $\tilde{\boldsymbol{n}}$ and $\tilde{\boldsymbol{n}}'$ are identical except that $\tilde{\boldsymbol{n}}'$ has one additional flow of type m' allocated to UPF k. Thus, $\tilde{\boldsymbol{n}}_j = \tilde{\boldsymbol{n}}'_j$, for $j \neq k$. Further, for $j \neq k$, $q_{jm} = q'_{jm}$, for all $m \in [M]$. Thus $\boldsymbol{u}_j = \boldsymbol{u}'_j$ for all $j \neq k$. Thus,

$$\tilde{\boldsymbol{n}}_j - \boldsymbol{u}_j = \tilde{\boldsymbol{n}}'_j - \boldsymbol{u}'_j.$$

For row k, let $\boldsymbol{e}_{\tilde{U}}$ and $\boldsymbol{e}_{\tilde{U}'}$ denote the one-hot vectors indicating the departing types. Let $\boldsymbol{e}_{m'}$ denote the one-hot vector with 1 in position m' (representing the additional flow of type m' in $\tilde{\boldsymbol{n}}'$). We want to show

$$\tilde{\boldsymbol{n}}_k - \boldsymbol{e}_{\tilde{U}} \leqslant \tilde{\boldsymbol{n}}_k + \boldsymbol{e}_{m'} - \boldsymbol{e}_{\tilde{U}'} \quad \text{element-wise.} \tag{21}$$

By Lemma 2, there are two cases to analyze: (i) $\tilde{U}' = \tilde{U}$ and (ii) $\tilde{U} \neq \tilde{U}'$ and $\tilde{U}' = m'$.

- If $\tilde{U}' = \tilde{U}$, the right hand side of Eq. (21) becomes

  $$\tilde{\boldsymbol{n}}_k + \boldsymbol{e}_{m'} - \boldsymbol{e}_{\tilde{U}'} = \tilde{\boldsymbol{n}}_k + \boldsymbol{e}_{m'} - \boldsymbol{e}_{\tilde{U}} = (\tilde{\boldsymbol{n}}_k - \boldsymbol{e}_{\tilde{U}}) + \boldsymbol{e}_{m'}.$$

  Since $\boldsymbol{e}_{m'}$ is a one-hot vector with all nonnegative entries, we have $(\tilde{\boldsymbol{n}}_k - \boldsymbol{e}_{\tilde{U}}) + \boldsymbol{e}_{m'} \geqslant \tilde{\boldsymbol{n}}_k - \boldsymbol{e}_{\tilde{U}}$ element-wise.
- If $\tilde{U}' \neq \tilde{U}$ and $\tilde{U}' = m'$, the right hand side of Eq. (21)

  $$\tilde{\boldsymbol{n}}_k + \boldsymbol{e}_{m'} - \boldsymbol{e}_{\tilde{U}'} = \tilde{\boldsymbol{n}}_k + \boldsymbol{e}_{m'} - \boldsymbol{e}_{m'} = \tilde{\boldsymbol{n}}_k.$$

  Since $\boldsymbol{e}_{\tilde{U}}$ is a one-hot vector (indicating one flow departed), $\tilde{\boldsymbol{n}}_k \geqslant \tilde{\boldsymbol{n}}_k - \boldsymbol{e}_{\tilde{U}}$ element-wise.

Thus the desired inequality holds for all rows of $\tilde{\boldsymbol{n}}, \tilde{\boldsymbol{n}}'$ and this completes the proof. □

Now we are ready to prove Theorem 2.

C. Proving Theorem 2

Proof of Theorem 2- Property 1): The result is an immediate consequence from the construction of the post-decision state. We provide the proof for the sake of completeness.

Recall the post-decision state value function given in Eq. (16). We observe that the right hand side of the equation depends only on $\tilde{\boldsymbol{n}}, \boldsymbol{u}, \boldsymbol{a}$, and the flow arrivals in the next state f'. However, it does not depend on f. Thus $\widetilde{V}(\tilde{\boldsymbol{n}}, f) = \widetilde{V}(\tilde{\boldsymbol{n}}, \cdot)$. This completes the proof of 1).

Henceforth, we denote the post-decision value function as $\widetilde{V}(\tilde{\boldsymbol{n}})$.

Proof of Theorem 2-Property 2): Recall the value iteration updates on the post-decision state

$$\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}) = \mathbb{E}_{\boldsymbol{u},f'}\Big[\min_{a\in\mathcal{A}(\tilde{\boldsymbol{n}}-\boldsymbol{u},f')}\Big\{\xi(\tilde{\boldsymbol{n}}-\boldsymbol{u},f') + \gamma\widetilde{V}_t(\tilde{\boldsymbol{n}}-\boldsymbol{u}+\boldsymbol{a})\Big\}\Big].$$

We show that each iterate $\widetilde{V}_t \in \mathcal{R}^{|\mathcal{S}|}$ is element-wise non-decreasing in $\tilde{\boldsymbol{n}}$ and the limit $\widetilde{V}^\star$ inherits this property. We prove by induction on the iteration count t.

a) Base case: $\widetilde{V}_0 \equiv \mathbf{0}$ is monotone by initialization.

b) Induction hypothesis: Assume for some t that

$$\tilde{\boldsymbol{n}} \leqslant \tilde{\boldsymbol{n}}' \quad \Rightarrow \quad \widetilde{V}_t(\tilde{\boldsymbol{n}}) \leqslant \widetilde{V}_t(\tilde{\boldsymbol{n}}')$$

c) Induction step: We now prove the result holds for iteration $t+1$ if it holds for the t^{th} iteration.

Construct a coupling of departures $(\tilde{U}, \tilde{U}')$ as in Lemma 2 for $\tilde{\boldsymbol{n}}_k$ and $\tilde{\boldsymbol{n}}'_k$. By Lemma 3, the pre-decision states satisfy

$$\boldsymbol{n} := \tilde{\boldsymbol{n}} - \boldsymbol{u} \;\leqslant\; \boldsymbol{n}' := \tilde{\boldsymbol{n}}' - \boldsymbol{u}' \quad \text{element-wise.} \tag{22}$$

Recall that for any pre-decision state matrix $\boldsymbol{n}$ and flow f',

$$V_t(\boldsymbol{n}, f') = \min_{a\in\mathcal{A}(\boldsymbol{n},f')}\big\{\xi(\boldsymbol{n},f') + \gamma\widetilde{V}_t(\boldsymbol{n}+\boldsymbol{a})\big\},$$

and the post-decision state value iteration,

$$\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}) = \mathbb{E}[V_t(\boldsymbol{n},f')] = \mathbb{E}[\min_{a\in\mathcal{A}(\boldsymbol{n},f')}\big\{\xi(\boldsymbol{n},f') + \gamma\widetilde{V}_t(\boldsymbol{n}+\boldsymbol{a})\big\}].$$

Since $\boldsymbol{n} \leqslant \boldsymbol{n}'$ element-wise (Eq. (22)), we have the following three facts.

(i) $\mathcal{A}(\boldsymbol{n}', f') \subseteq \mathcal{A}(\boldsymbol{n}, f')$. Thus, the minimization over the feasible set $\mathcal{A}(\boldsymbol{n}', f')$ cannot yield a lower value than the minimization over the set $\mathcal{A}(\boldsymbol{n}, f')$.

(ii) The cost function ξ is non-decreasing in the number of flows by definition. Thus

$$\boldsymbol{n} \leqslant \boldsymbol{n}' \quad \Rightarrow \quad \xi(\boldsymbol{n},f') \leqslant \xi(\boldsymbol{n}',f').$$

(iii) For a given action matrix $\boldsymbol{a}$, using the induction hypothesis, we have

$$\widetilde{V}_t(\boldsymbol{n}+\boldsymbol{a}) \leqslant \widetilde{V}_t(\boldsymbol{n}'+\boldsymbol{a}).$$

Consider $a \in \mathcal{A}(\boldsymbol{n}', f')$. By (i), a is a feasible action at $(\boldsymbol{n}, f')$. Using properties (ii) and (iii),

$$\xi(\boldsymbol{n},f') + \gamma\widetilde{V}_t(\boldsymbol{n}+\boldsymbol{a}) \;\leqslant\; \xi(\boldsymbol{n}',f') + \gamma\widetilde{V}_t(\boldsymbol{n}'+\boldsymbol{a}). \tag{23}$$

Since $\mathcal{A}(\boldsymbol{n}', f') \subseteq \mathcal{A}(\boldsymbol{n}, f')$, applying minimization over $\mathcal{A}(\boldsymbol{n}, f')$ on the left hand side of Eq. (23) and over $\mathcal{A}(\boldsymbol{n}', f')$ on the right hand side of Eq. (23) gives

$$\begin{aligned} V_t(\boldsymbol{n},f') &= \min_{a\in\mathcal{A}(\boldsymbol{n},f')}[\xi(\boldsymbol{n},f') + \gamma\widetilde{V}_t(\boldsymbol{n}+\boldsymbol{a})] \\ &\leqslant \min_{a\in\mathcal{A}(\boldsymbol{n}',f')}[\xi(\boldsymbol{n}',f') + \gamma\widetilde{V}_t(\boldsymbol{n}'+\boldsymbol{a})] = V_t(\boldsymbol{n}',f'). \end{aligned} \tag{24}$$

From PDS construction, we know that,

$$\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}) = \mathbb{E}[V_t(\boldsymbol{n},f')], \quad \widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}') = \mathbb{E}[V_t(\boldsymbol{n}',f')].$$

Using Eq. (24), Lemma 3, and Proposition 1 we have

$$\widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}) \leqslant \widetilde{V}_{t+1}(\tilde{\boldsymbol{n}}').$$

By induction, each $\widetilde{V}_t(\tilde{\boldsymbol{n}}) \leqslant \widetilde{V}_t(\tilde{\boldsymbol{n}}')$ is non-decreasing for all t. Since $\widetilde{V}_t \to \widetilde{V}^\star$ as $t \to \infty$, $\widetilde{V}^\star(\tilde{\boldsymbol{n}}) \leqslant \widetilde{V}^\star(\tilde{\boldsymbol{n}}')$. This completes the proof of 2).

Proof of Theorem 2- Property 3): Given the optimal post-decision value vector $\widetilde{V}^\star$, the optimal pre-decision value is

$$V^\star(\boldsymbol{n},f) = \min_{a\in\mathcal{A}(\boldsymbol{n},f)}\big\{\xi(\boldsymbol{n},f) + \gamma\widetilde{V}^\star(\boldsymbol{n}+\boldsymbol{a})\big\}.$$

By Property 2), $\widetilde{V}^\star(\cdot)$ is non-decreasing. We know that

$$\boldsymbol{n} \leqslant \boldsymbol{n}' \Rightarrow \widetilde{V}^\star(\boldsymbol{n}+\boldsymbol{a}) \leqslant \widetilde{V}^\star(\boldsymbol{n}'+\boldsymbol{a}).$$

Using the monotonicity of $\widetilde{V}^\star(\cdot)$ and (i) – (iii) from the proof of Property 2), we get

$$\begin{aligned} V^\star(\boldsymbol{n},f) &= \min_{a\in\mathcal{A}(\boldsymbol{n},f)}[\xi(\boldsymbol{n},f) + \gamma\widetilde{V}^\star(\boldsymbol{n}+\boldsymbol{a})] \\ &\leqslant \min_{a\in\mathcal{A}(\boldsymbol{n}',f)}[\xi(\boldsymbol{n}',f) + \gamma\widetilde{V}^\star(\boldsymbol{n}'+\boldsymbol{a})] = V^\star(\boldsymbol{n}',f). \end{aligned}$$

This completes the proof of Property 3). □

IX. Proofs of Theorem 1 and Theorem 3

Proof of Theorem 1: Let $\mathcal{T} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ be defined as,

$$\mathcal{T}(\widetilde{V})(\tilde{s}) = \sum_{f',\boldsymbol{u}} \mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}})\cdot\mathbb{P}(f') \min_{a'\in\mathcal{A}(s')}\Big[\xi(\tilde{\boldsymbol{n}}-\boldsymbol{u},f') + \gamma\widetilde{V}(\tilde{\boldsymbol{n}}-\boldsymbol{u}+\boldsymbol{a}',f')\Big]. \tag{25}$$

Define $\widehat{\mathcal{T}}$ as the sampled operator for some observed f' and $\boldsymbol{u}$ at time t, then

$$\widehat{\mathcal{T}}(\widetilde{V}_t)(\tilde{s}) = \min_{a'\in\mathcal{A}(s')}\Big[\xi(\tilde{\boldsymbol{n}}-\boldsymbol{u},f') + \gamma\widetilde{V}_t(\tilde{\boldsymbol{n}}-\boldsymbol{u}+\boldsymbol{a}',f')\Big].$$

We can rewrite Eq. (17) in terms of $\mathcal{T}$ and $\widehat{\mathcal{T}}$ as

$$\widetilde{V}_{t+1}(\tilde{s}) = \widetilde{V}_t(\tilde{s}) + \alpha_t\Big(\mathcal{T}(\widetilde{V}_t)(\tilde{s}) - \widetilde{V}_t(\tilde{s}) + \epsilon_{t+1}(\tilde{s})\Big),$$

where, $\epsilon_{t+1}(\tilde{s}) = \widehat{\mathcal{T}}(\widetilde{V}_t)(\tilde{s}) - \mathcal{T}(\widetilde{V}_t)(\tilde{s})$. Rearranging,

$$\widetilde{V}_{t+1}(\tilde{s}) - \widetilde{V}_t(\tilde{s}) = \alpha_t\Big(\mathcal{T}(\widetilde{V}_t)(\tilde{s}) - \widetilde{V}_t(\tilde{s}) + \epsilon_{t+1}(\tilde{s})\Big). \tag{26}$$

Let $\mathcal{F}_t = \{(s_{t'}, a_{t'}, \xi_{t'}, s'_{t'})\}_{0\leqslant t'\leqslant t}$ represent the information up to time t. It follows that $\{\epsilon_{t+1}(\tilde{s})\}_{t\geqslant 0}$ forms a martingale difference sequence and $\mathbb{E}[\epsilon_{t+1}(\tilde{s})|\mathcal{F}_t] = 0$. The iterates in Eq. (26) are equivalent to the discretized version of the ordinary differential equation (ODE), $\dot{\widetilde{V}} = \mathcal{T}(\widetilde{V}) - \widetilde{V}$. From

[28], it can be argued that the convergence of these updates is equivalent to the convergence of the aforementioned ODE. Also, $\mathcal{T}$ is a γ-contraction under the sup-norm i.e.,

$$||\mathcal{T}(f) - \mathcal{T}(g)||_\infty \leqslant \gamma ||f - g||_\infty \quad \forall f, g.$$

Thus the iterates $\widetilde{V}_t$ converge to the optimal value $\widetilde{V}^\star$. □

Proof of Theorem 3: The first-stage update, $\widetilde{Q}(\tilde{s}, \boldsymbol{a})$ is of the same form as the post-decision state value iteration update in Eq. (17), but indexed by state-action pairs instead of just states. For each fixed $(\tilde{s}, \boldsymbol{a})$, the update

$$\widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a}) = \widetilde{Q}_t(\tilde{s}, \boldsymbol{a}) + \alpha_t \Big[\gamma \min_{\boldsymbol{a}'} Q_t(s', \boldsymbol{a}') - \widetilde{Q}_t(\tilde{s}, \boldsymbol{a})\Big]$$

is a stochastic approximation of the operator

$$T\widetilde{Q}(\tilde{s}, \boldsymbol{a}) := \sum_{s'} \mathbb{P}(\boldsymbol{n}'|\tilde{\boldsymbol{n}}) \cdot \mathbb{P}(f')[\gamma \min_{\boldsymbol{a}'} Q(s', \boldsymbol{a}')],$$

which can be verified as a γ-contraction. Therefore, using the same ODE-based argument as in Theorem 1, it is guaranteed that the iterates $\widetilde{Q}_t(\tilde{s}, \boldsymbol{a})$ converge almost surely to its fixed point $\widetilde{Q}^\star(\tilde{s}, \boldsymbol{a})$.

The cost function ξ is constant for a given state, so the second-stage update $Q_{t+1}(s, \boldsymbol{a}) = \xi(s) + \widetilde{Q}_{t+1}(\tilde{s}, \boldsymbol{a})$ is a deterministic function of $\widetilde{Q}$, hence, the Q-table also converges almost surely. This completes the proof. □

X. Conclusion

In this work, we studied the delay- and energy-aware flow allocation problem in wireless systems under resource constraints. We formulated the problem as an MDP and proposed online RL algorithms based on post-decision state (PDS) learning. By leveraging the decomposable structure of the system dynamics, i.e., separating the controllable and exogenous factors, our approach enables efficient learning with faster convergence. We proved key structural properties of the PDS approach and demonstrated how they can be exploited to accelerate learning and improve convergence. We proved the convergence of the proposed algorithm and evaluated its performance against the standard Q-learning algorithm, validating its effectiveness. We also proposed a scalable empirical Tournament method that utilizes the structural properties of the formulation to provide an efficient solution in real-time and we verified its performance against a Greedy algorithm. Our simulation results validated the accelerated convergence and improved cost performance of the proposed algorithms.

References

[1] X. Wu, J. Farooq, and J. Chen, "Joint admission control and resource provisioning for URLLC traffic in O-RAN: A constrained multi-agent reinforcement learning approach," in *IEEE International Conference on Communications (ICC)*, 2025, pp. 1426–1431.

[2] A. Vatankhah and R. Liscano, "QoS-aware energy-efficient time-slotted channel hopping scheduling algorithm," in *IEEE International Conference on Communications (ICC)*, 2025, pp. 3538–3544.

[3] C. Wang, X. Ma, R. Xing, S. Li, A. Zhou, and S. Wang, "Delay-and resource-aware satellite UPF service optimization," *IEEE Transactions on Mobile Computing*, 2024.

[4] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on selected areas in communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[5] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5G hetnet," *IEEE Journal on selected areas in communications*, vol. 38, no. 12, pp. 2773–2782, 2020.

[6] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1529–1541, 2021.

[7] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, 2019.

[8] Z. Han and K. R. Liu, *Resource allocation for wireless networks: basics, techniques, and applications*. Cambridge university press, 2008.

[9] S. Stańczak, M. Wiczanowski, and H. Boche, *Resource allocation in wireless networks: theory and algorithms*. Springer Science & Business Media, 2006, vol. 4000.

[10] T. D. Braun, H. J. Siegel, A. A. Maciejewski, and Y. Hong, "Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions," *Journal of Parallel and Distributed Computing*, vol. 68, no. 11, pp. 1504–1516, 2008.

[11] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 107–131, 1999.

[12] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction," *Journal of network and Computer Applications*, vol. 34, no. 4, pp. 1068–1077, 2011.

[13] Z. Xu, L. Zhou, H. Dai, W. Liang, W. Zhou, P. Zhou, W. Xu, and G. Wu, "Energy-aware collaborative service caching in a 5G-enabled mec with uncertain payoffs," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1058–1071, 2022.

[14] S. Huang, B. Lv, R. Wang, and K. Huang, "Scheduling for mobile edge computing with random user arrivals—an approximate mdp and reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7735–7750, 2020.

[15] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efficient wireless communication," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 6262–6266, 2011.

[16] F. Fu and M. van der Schaar, "Structural solutions for dynamic scheduling in wireless multimedia transmission," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 5, pp. 727–739, 2012.

[17] J. Yang, S. Zhang, X. Wu, Y. Ran, and H. Xi, "Online learning-based server provisioning for electricity cost reduction in data center," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 1044–1051, 2017.

[18] X. Chen, Z. Han, H. Zhang, G. Xue, Y. Xiao, and M. Bennis, "Wireless resource scheduling in virtualized radio access networks using stochastic learning," *IEEE Transactions on Mobile Computing*, vol. 17, no. 4, pp. 961–974, 2018.

[19] X. He, R. Jin, and H. Dai, "Deep pds-learning for privacy-aware offloading in mec-enabled iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4547–4555, 2019.

[20] M. Min, X. Wan, L. Xiao, Y. Chen, M. Xia, D. Wu, and H. Dai, "Learning-based privacy-aware offloading for healthcare iot with energy harvesting," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4307–4316, 2019.

[21] M. G. Bhat, S. Moothedath, and P. Chaporkar, "Post-decision state-based online learning for delay-energy-aware flow allocation in wireless systems," *Submitted to IEEE International Conference on Communications*, 2026, arXiv:2601.03108.

[22] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

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

[24] N. Salodkar, A. Bhorkar, A. Karandikar, and V. S. Borkar, "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 732–742, 2008.

[25] J. Zhang, X. He, and H. Dai, "Blind post-decision state-based reinforcement learning for intelligent iot," *IEEE Internet of Things Journal*, vol. 10, no. 12, pp. 10 605–10 620, 2023.

[26] S. Roch, *Modern Discrete Probability: An Essential Toolkit*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2024.

[27] T. Lindvall, "On strassen's theorem on stochastic domination," 1999.

[28] V. S. Borkar and S. P. Meyn, "The ODE method for convergence of stochastic approximation and reinforcement learning," *SIAM Journal on Control and Optimization*, vol. 38, no. 2, pp. 447–469, 2000.