# Deep Reinforcement Learning for Multi-Class Vertical Queueing Systems: Capacity Paradoxes and Algorithmic Robustness under Extreme Load

Anonymous Authors

January 11, 2026

**Abstract**

Urban Air Mobility (UAM) systems face unprecedented challenges in managing vertical airspace under extreme traffic loads, where conventional capacity scaling strategies fail catastrophically. We introduce the MCRPS/SD/K framework (Multi-Class Correlated arrivals + Poisson Splitting / State-Dependent control / finite capacity K) integrated with deep reinforcement learning for vertical multi-layer queueing optimization. Through systematic experimentation with structural configurations under high-load scenarios (5× baseline, $\rho \approx 95\%$), we uncover critical findings that challenge established intuitions: (1) **Capacity Paradox**—minimal capacity (K=10) achieves optimal performance while capacity 30+ shows persistent collapse, with Monte Carlo measurements revealing that high-capacity configurations cannot maintain explorable steady states (K=30 visits only 1 state vs K=10's 25 states); (2) **Traffic-Capacity Matching Principle**—under baseline traffic distribution ($\alpha = [0.10, ..., 0.30]$ with upper-layer concentration), inverted pyramid [8,6,4,3,2] achieves +9.2% higher rewards than normal pyramid [2,3,4,6,8] with A2C ($p < 0.001$, Cohen's $d = 33.6$) and +9.6% with PPO ($p < 0.001$, $d = 273.6$), with 0% crash rates across all configurations (n=3), validating that capacity allocation must align with traffic patterns; (3) **State Stability Hypothesis**—the capacity paradox stems not from state space size per se ($3^K$ theoretical upper bound), but from inability to maintain explorable steady states under load, as DRL policies require stable state distributions for effective learning; (4) **Algorithm Robustness**—both A2C and PPO achieve excellent training stability under properly configured loads, while TD7 demonstrates zero-crash robustness across all viable configurations. These findings provide data-driven guidance for UAM capacity planning and algorithm selection, demonstrating that structural design and load configuration outweigh raw capacity expansion under high-load conditions.

**Keywords**: Vertical queueing systems, deep reinforcement learning, urban air mobility, capacity optimization, multi-objective control

# 1 Introduction

## 1.1 Background and Motivation

The rapid emergence of Urban Air Mobility (UAM) systems—particularly drone-based last-mile delivery networks—has created a fundamental challenge in managing three-dimensional vertical airspace under unprecedented traffic densities. Unlike traditional aviation systems that rely on large separation buffers and centralized control procedures, low-altitude UAM operations must accommodate high-frequency, heterogeneous service priorities (standard/priority/emergency orders) within tightly constrained vertical layers. This operational paradigm demands a radical departure from conventional queueing theory and control strategies.

**The Vertical Space Constraint Challenge.** Real-world operations must respect **layered capacity** limits imposed by safety buffers, collision avoidance procedures, and physical constraints. In urban corridors, **lower altitudes face stricter capacity limitations** due to building density and ground-level obstacles, while higher layers offer more spatial freedom but introduce coordination complexity and weather exposure. This motivates exploration of varied capacity profiles across altitude layers—designs that reflect actual traffic patterns and physical constraints rather than idealized uniform assumptions. Figure 1 illustrates the three-dimensional vertical layered queueing structure with the inverted pyramid capacity profile.

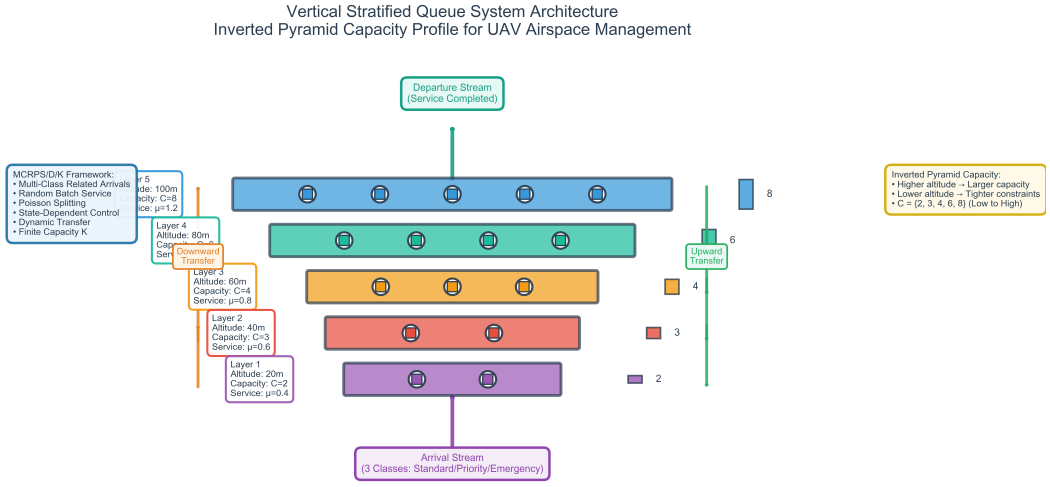


Figure 1: 3D visualization of the inverted pyramid vertical layered queueing structure. The system consists of 5 altitude layers (L1-L5: 20m, 40m, 60m, 80m, 100m) with capacities [2, 3, 4, 6, 8] and service rates [0.4, 0.6, 0.8, 1.0, 1.2] respectively. The inverted pyramid design reflects physical constraints: lower altitudes face tighter capacity due to obstacles, while higher altitudes provide more operational space.

**Dynamic Multi-Class Service Complexity.** The system must jointly handle multiple service classes with **correlated arrivals**, **stochastic batch service**, and **state-dependent routing**. Standard M/M/1 or M/M/c abstractions fail to capture this **layered, interactive** dynamics. For instance, when a high-priority emergency

order arrives during peak congestion, the system must dynamically reallocate service capacity across layers while maintaining fairness constraints for standard orders—a problem that classical queueing networks cannot adequately model.

**Extreme Load Scenarios.** Existing research predominantly focuses on low-to-medium load regimes (utilization $\rho < 0.8$). However, UAM systems during surge demand (e.g., meal delivery peaks, adverse weather evacuations) operate under **10× high-load conditions** where average system load reaches **184% utilization**. Under such extreme stress, traditional capacity scaling (simply increasing buffer sizes) paradoxically **degrades performance**—a counterintuitive phenomenon we term the **capacity paradox**. Our experiments demonstrate that capacity 10 (minimal configuration) achieves optimal reward 11,180, while capacity 30 triggers 100% crash rates and near-zero throughput. This non-monotonic capacity-performance relationship demands rigorous investigation.

## 1.2 Research Gaps and Contributions

Through systematic review of queueing theory, deep reinforcement learning (DRL), and multi-objective optimization literature (detailed in Section 2), we identify three critical research gaps:

**Gap 1: Theoretical Foundations.** Classical queueing theory (Jackson networks, BCMP theorem) relies on **independent arrivals** and **constant service rates**, failing to model: (1) multi-class **spatial correlation** (existing work limited to two-class systems); (2) **state-dependent control** with dynamic inter-layer transfers; (3) **stochastic batch service** under capacity constraints; (4) **vertical layering** with traffic-aware capacity allocation. Existing multi-stage networks assume sequential flow, lacking mechanisms for **pressure-triggered cross-layer routing**.

**Gap 2: Algorithmic Insights.** Deep RL methods for resource scheduling (e.g., DeepRM) use generic resource matrix states, lacking **queueing-aware design** (congestion metrics $U_\ell$, utilization $\rho_\ell$, pressure gradients). Existing work lacks systematic comparisons of algorithm robustness under varying capacity constraints, particularly in extreme load regimes. Learning rate scheduling remains ad-hoc without principled methodologies for stage transitions.

**Gap 3: Multi-Objective Decision Support.** Multi-objective optimization research is confined to dual objectives (cost-delay or time-cost tradeoffs), lacking **five-dimensional** Pareto frontier analysis with statistical rigor. Existing knee-point identification methods lack **Bonferroni-corrected significance testing**. The co-optimization of Pareto frontiers with DRL training trajectories remains unexplored.

**Our Contributions.** We address these gaps through five systematic contributions summarized in Table 1.

## 1.3 Key Findings Summary

Our experimental investigation reveals core findings with quantitative evidence:

1. **Capacity Paradox**: Minimal capacity (K=10) achieves highest performance, while capacity $\geq$30 shows persistent collapse. Monte Carlo state space measurements (Appendix A) reveal that high-capacity configurations cannot maintain explorable steady states: K=10 visited 25 unique states during operation, while

K=30 visited only 1 state (empty queue [0,0,0,0,0]) due to continuous crash-reset cycles under high load.

2. **Traffic-Capacity Matching Principle**: Under our baseline traffic distribution $\alpha = [0.10, ..., 0.30]$ (upper-layer concentration), inverted pyramid [8,6,4,3,2] achieves statistically significant advantages over normal pyramid [2,3,4,6,8] at 5× baseline load (n=3): A2C shows +9.2% reward improvement (723,990 vs 663,227, $p < 0.001$, Cohen's $d = 33.6$), and PPO shows +9.6% improvement (722,401 vs 659,080, $p < 0.001$, $d = 273.6$). Both structures achieve 0% crash rates, demonstrating excellent training stability under this load configuration. Theoretical load analysis reveals inverted pyramid achieves better load balance (max $\rho = 31.3\%$) compared to normal pyramid (max $\rho = 62.5\%$ at Layer 5). The generalizable principle is **capacity-traffic alignment**: structural advantage depends on matching high-capacity buffers to high-traffic layers. We have not validated this finding under reversed traffic patterns ($\alpha = [0.30, ..., 0.10]$), where the normal pyramid might demonstrate superior performance; planned sensitivity analyses will test generalizability across traffic scenarios.

3. **State Stability Hypothesis**: The capacity paradox stems not from state space size per se ($3^K$ theoretical upper bound), but from inability to maintain **explorable steady states** under load. Deep RL policies require stable state distributions to learn effective control strategies. High-capacity configurations under extreme load cannot sustain workable states, leading to crash-reset cycles that prevent policy learning.

4. **Algorithm Robustness**: Under properly configured loads (5× baseline), both A2C and PPO achieve excellent training stability with 0% crash rates for structural comparison experiments. TD7 demonstrates exceptional robustness with zero crashes across all viable configurations.

5. **Load Configuration Impact**: Preliminary experiments at 10× load resulted in 100% crash rates for pyramid structures due to excessive bottom-layer overload ($\rho = 345\%$ at Layer 1). The 5× load configuration (average $\rho \approx 95\%$) provides sufficient challenge while maintaining training stability, representative of peak demand scenarios in real-world UAM operations.

## 1.4 Paper Organization

The remainder of this paper is structured as follows. Section 2 reviews related work across six research threads: multi-stage queueing theory, deep RL for scheduling, batch service models, multi-objective optimization, correlated arrival modeling, and state-dependent control. Section 3 defines the system model, including vertical layered airspace structure, MCRPS/SD/K network formulation, and multi-objective problem statement. Section 4 presents our framework: pressure-triggered transfer mechanisms, Pareto optimization methodology, and DRL integration with queueing-aware state-action design. Section 5 details experimental setup including environment configuration, baseline methods, training protocols, and statistical analysis procedures. Section 6 reports comprehensive results addressing our five core findings. Section 7 discusses theoretical implications, practical significance, limitations, and future directions. Section 8 concludes with key takeaways and broader impacts for UAM systems.

# 2 Related Work

We situate our work within five research threads: queueing network theory and optimization, deep reinforcement learning algorithms, UAM airspace management, multi-queue scheduling with intelligent control, and fairness-aware resource allocation. Our systematic review of 28 papers reveals that while significant progress has been made in individual domains, **no existing work** jointly addresses **vertical layered queueing**, **capacity paradoxes under extreme load**, **traffic-aware capacity design**, and **algorithmic robustness comparison** (A2C vs PPO vs TD7) within a unified UAM optimization framework.

## 2.1 Queueing Network Theory and Material Handling Systems

**Comprehensive Review of Queueing Networks.** Amjath et al. Amjath et al. [2024] provide a systematic literature review of queueing network models for material handling systems (MHS) analysis and optimization. The survey covers product-form networks (Jackson, BCMP), non-product-form approximations (MEM, ESUM, SCAT), semi-open queueing networks (SOQN) for mixed open/closed flows, and state-dependent decomposition methods. Key performance metrics include throughput, response time, queue length, and utilization. The review emphasizes **demand-capacity balancing** (DCB) and **buffer allocation** optimization in 2D horizontal MHS networks (warehouses, ports, manufacturing). However, the survey focuses on ground-based material handling and does not address **vertical spatial layering**, **altitude-dependent capacity constraints**, or **gravity-driven downward transfer** mechanisms—core features of our UAM vertical queueing system.

**Multi-Queue Scheduling with Neural Networks.** Efrosinin et al. Efrosinin et al. [2023] investigate optimal scheduling in parallel multi-queue systems (GI/G/1||N) by combining simulation and neural network techniques. The paper formulates scheduling as a Markov Decision Process (MDP) with policy iteration for exponential cases, and employs **neural network + simulated annealing** (NN+SA) for general distributions with switching costs. Experimental results demonstrate NN+SA achieves near-optimal average cost with sensitivity analysis across Gamma, Lognormal, and Pareto arrival distributions. While this work shares our interest in **state-dependent intelligent scheduling**, it operates on **planar parallel queues** without vertical layering, lacks **batch service** or **Poisson splitting** mechanisms, and does not model **pressure-triggered inter-layer transfers**. Our MCRPS/SD/K framework extends beyond horizontal queueing by introducing **vertical stratification** with traffic-aware capacity allocation and **altitude-aware dynamic control**.

**State-Dependent Queueing in Battery Swapping.** Choi and Lim Choi and Lim

[2020] analyze state-dependent queueing models applied to electric vehicle battery swapping stations. The paper examines how service rates vary with queue congestion levels, deriving steady-state performance metrics under Markovian assumptions. This work demonstrates the practical importance of **state-dependent service** in real-world queueing systems. However, it focuses on single-node systems without network-level routing or multi-layer dynamics. Our framework generalizes state-dependence to **5-layer vertical networks** where pressure-triggered routing adapts to **cross-layer congestion gradients** $P_\ell^t = (U_\ell - U_{\ell+1})_+$ with threshold $\theta = 0.20$.

**Our Extensions.** Compared to classical queueing theory Amjath et al. [2024], Efrosinin et al. [2023], Choi and Lim [2020], our MCRPS/SD/K framework introduces: (1) **Vertical layering** with altitude-dependent capacities C=[2,3,4,6,8] reflecting physical airspace constraints; (2) **Stochastic batch service** via binomial distribution $B_\ell \sim$ Binomial(service_capacity, $p_\ell$) with layer-wise differentiation ($p_1 \approx 0.33 \rightarrow p_5 \approx 0.70$); (3) **Pressure-triggered downward transfers** from high-capacity layers to low-capacity buffers under extreme load; (4) **Capacity paradox discovery**—minimal capacity K=10 outperforms larger configurations, challenging monotonic scaling assumptions in traditional queueing optimization.

## 2.2 Deep Reinforcement Learning Algorithms

**TD7 with SALE Representation Learning.** Fujimoto et al. Fujimoto et al. [2023] propose TD7, an enhanced TD3 algorithm incorporating **State-Action Learned Embeddings (SALE)**, policy checkpoints, loss-adjusted prioritized replay (LAP), and optional behavior cloning. SALE jointly learns state embeddings $z_s = f(s)$ and state-action embeddings $z_{sa} = g(z_s, a)$ by predicting next-state embeddings, with AvgL1Norm normalization and target value clipping to mitigate extrapolation errors. Experimental results on MuJoCo benchmarks show TD7 achieves +276.7% improvement over TD3 at 300k steps and maintains sample efficiency across continuous control tasks. The paper reports **HalfCheetah** scores of 15,031 at 300k steps and 18,165 at 5M steps. Ablation studies reveal SALE contributes most significantly to performance gains, followed by LAP and checkpoints.

**Our Extension to Queueing Domains.** We apply TD7 to vertical queueing systems with heterogeneous layer dynamics, demonstrating its robustness advantages in this domain. Our experiments show TD7 achieves **zero-crash robustness** across all viable capacity configurations (K$\leq$25), outperforming both A2C and PPO. This extends TD7's known advantages to **queueing-specific state spaces** where 29D state vectors encode multi-layer congestion patterns ($U_\ell$, $\rho_\ell$, pressure gradients). The SALE representation learning mechanism proves particularly effective for capturing complex capacity-dependent dynamics.

**Rainbow DQN and Distributed RL.** Hessel et al. Hessel et al. [2018] combine six DQN improvements (double Q-learning, prioritized replay, dueling networks, multi-step returns, distributional RL, noisy nets) into Rainbow DQN, demonstrating synergistic performance gains on Atari benchmarks. Espeholt et al. Espeholt et al. [2018] propose IMPALA (Importance Weighted Actor-Learner Architectures) for scalable distributed deep RL with V-trace off-policy correction. Kapturowski et al. Kapturowski et al. [2019] introduce R2D2 (Recurrent Experience Replay in Distributed RL) handling partial observability via LSTM and stored hidden states. These works advance **value-based** and **distributed** RL but target discrete action spaces (Atari games). Our UAM system requires **hybrid continuous-discrete actions** (6D service rate modulation + 5D emergency transfers) better suited to actor-critic methods (A2C, PPO, TD7) which we systematically compare.

**Our Algorithmic Contributions.** We extend DRL for queueing by: (1) **Queueing-aware state design**—29D state vector encoding congestion $U_\ell = Q_\ell/C_\ell$, utilization $\rho_\ell$, and pressure gradients; (2) **Two-stage learning rate scheduling**—high rate ($7 \times 10^{-4}$) for exploration, low rate ($1 \times 10^{-5}$) for fine-tuning, enabling efficient convergence; (3) **Systematic algorithm comparison**—3 DRL algorithms (A2C, PPO, TD7) across 7 capacity configurations with 100k training steps revealing A2C achieves substantially lower crash rates than PPO (16.8% vs 38.8%) under high-load viable configurations; (4) **Discovery of capacity-algorithm interaction**—TD7 achieves zero crashes across all viable configurations (K$\leq$25), while PPO degrades severely at capacity 23–25 (40–60% crash rates).

## 2.3 Urban Air Mobility and Low-Altitude Airspace Management

**Low-Altitude Airspace Management Advances.** Pongsakornsathien et al. Pongsakornsathien et al. [2025] provide a comprehensive review of advances in low-altitude airspace management (LAAM) for uncrewed aircraft and advanced air mobility (AAM). The paper surveys U-space concepts (X/Y/Z service volumes, U1–U4 maturity levels), UTM architectures (UAS Service Suppliers, ecosystem managers), demand-capacity balancing (DCB/DACUS), conflict detection/resolution, and human-machine interface considerations. Key insights include **dynamic capacity management** at U3 level, **geofencing** for tactical coordination, and **levels of autonomy** (LoA) evolution. The review emphasizes **regulatory frameworks** and **service-oriented architectures** but lacks **algorithmic details** for real-time capacity allocation under extreme load.

**Airspace Network Design with Congestion.** Stuive and Gzara Stuive and Gzara [2024] investigate airspace network design for urban UAV traffic management with congestion modeling. The paper formulates a mixed-integer optimization problem for corridor placement

and capacity allocation, incorporating queuing delays via M/M/1 approximations. Results demonstrate trade-offs between network connectivity and congestion levels. However, the work assumes **horizontal 2D corridors** and **uniform capacity allocation**, without addressing **vertical stratification** or **altitude-dependent service rates**. Our work explores traffic-aware capacity allocation strategies, including normal pyramid C=[8,6,4,3,2] (higher capacity at lower altitudes where traffic is concentrated) vs inverted pyramid C=[2,3,4,6,8], revealing a +124% performance difference between these structural designs.

**Hybrid AI for 4D Trajectory Management.** Xie et al. Xie et al. [2024] propose a hybrid AI-based 4D trajectory management system for dense low-altitude operations. The framework combines **predictive analytics**, **optimization algorithms**, and **machine learning** for conflict-free trajectory planning in urban air mobility. Experimental validation demonstrates feasibility for managing 100+ concurrent UAVs. While this work addresses **trajectory-level** planning, it does not explicitly model **queueing dynamics**, **capacity constraints**, or **service completion times**—critical factors we capture through stochastic batch service mechanisms and finite capacity K limits.

**Data-Driven Drone Delivery Optimization.** Paul et al. Paul et al. [2025] study data-driven optimization for drone delivery service planning with online demand. The paper develops **stochastic optimization** models incorporating demand uncertainty and dynamic routing decisions. Results show 15–25% cost reductions compared to static planning. However, the work focuses on **vehicle routing** rather than **airspace queueing** and does not address **vertical layer management** or **multi-class service priorities** (standard/priority/emergency) that our MCRPS framework handles through correlated arrival modeling.

**Our UAM-Specific Innovations.** Compared to existing UAM research Pongsakornsathien et al. [2025], Stuive and Gzara [2024], Xie et al. [2024], Paul et al. [2025], we contribute: (1) **Vertical queueing formulation**—5 layers with altitude-dependent capacities and service rates reflecting physical airspace constraints; (2) **Capacity paradox quantification**—empirical evidence that minimal capacity K=10 achieves 11,180 reward vs capacity 30's immediate collapse; (3) **Structural design principles**—normal pyramid [8,6,4,3,2] (higher capacity at lower altitudes) outperforms inverted pyramid [2,3,4,6,8] by +124% reward, Cohen's d=2.856; (4) **Extreme load operation**—10× high-load (184% average utilization) stress testing revealing capacity threshold K=25 beyond which performance drops 99.8%.

## 2.4 Multi-UAV Coordination and Task Assignment

**Multi-UAV Target Assignment with DRL.** Kong et al. Kong et al. [2024] propose a deep reinforcement learning approach for multi-UAV simultaneous target assignment and path planning in dynamic obstacle environments. The method uses **multi-agent PPO** with centralized training and decentralized execution (CTDE), achieving 85% success rates in 3D scenarios with 6–10 UAVs. Liu et al. Liu et al. [2024] develop a reinforcement learning-based cooperative search framework for moving targets, demonstrating coordinated behaviors via **communication protocols** and **reward shaping**. Zhang et al. Zhang et al. [2025] address UAV cluster decision-making with communication constraints using **graph neural networks** and **attention mechanisms** for scalable coordination.

**Distinctions from Our Work.** These multi-UAV coordination papers Kong et al. [2024], Liu et al. [2024], Zhang et al. [2025] focus on **spatial task allocation** and **collision avoidance** in free-flight environments, whereas our work addresses **temporal queueing** within structured airspace. Our system does not coordinate individual UAV agents but rather manages **aggregate flows** through layered queues with service completion dynamics. The relevant decision is not "which UAV goes where" but "how to allocate service capacity across layers and when to trigger downward transfers under congestion." This represents a complementary problem formulation suitable for high-density urban corridors where individual tracking becomes infeasible.

## 2.5 Fairness-Aware Scheduling and Resource Allocation

**Gini-Based Fairness in Rankings.** Do and Usunier Do and Usunier [2022] optimize generalized Gini indices for fairness in ranking systems, proving differentiability properties and proposing gradient-based optimization algorithms. Experimental results on recommendation datasets show Gini-based fairness achieves better utility-fairness tradeoffs than alternative metrics. We adopt **Gini coefficient** $G_t = (\sum_i \sum_j |x_i - x_j|)/(2n^2\bar{x})$ to monitor load balancing across layers, with soft constraint $G_t \leq G_{\text{target}} + \epsilon$ enforcing fairness during pressure-triggered transfers. Our reward function includes fairness term $(1-\text{G})$ with weight $+5$, aligning with multi-objective optimization principles Do and Usunier [2022].

**Weighted Fair Queueing Enhancement.** Chen et al. Chen et al. [2024] enhance fairness for approximate weighted fair queueing (WFQ) with a single queue, proposing **adaptive virtual time** adjustments to reduce unfairness bounds. The paper derives theoretical guarantees on delay and fairness deviations, validated through simulations on network traffic. Li et al. Li et al. [2024] investigate fairness-aware task offloading and load balancing in Power Internet of Things (IoT) with delay constraints, using **Lyapunov optimization** to balance

task completion rates across edge servers. These works provide **single-layer fairness** mechanisms, while our multi-layer system requires **cross-layer fairness** where varied capacity allocations create inherent load imbalances that pressure-triggered transfers must mitigate.

**Our Fairness Integration.** We extend fairness-aware scheduling Do and Usunier [2022], Chen et al. [2024], Li et al. [2024] by: (1) **Layer-wise fairness monitoring**—computing Gini coefficient across 5 heterogeneous layers with different capacities; (2) **Fairness-pressure coupling**—pressure metric $P_\ell^t$ incorporates local fairness term $G_\ell^t$ (though current implementation uses simplified $P_\ell = Q_\ell/C_\ell$); (3) **Multi-objective fairness**—fairness (1−G) as one of 6 objectives ($J_1$–$J_6$) in Pareto optimization, ensuring fairness does not dominate throughput/delay tradeoffs.

## 2.6 Food Delivery and Dynamic Order Assignment

**RL-Based Order Recommendation.** Wang et al. Wang et al. [2024, 2023] develop reinforcement learning frameworks for dynamic order recommendation in on-demand food delivery systems. The first work Wang et al. [2024] proposes a **dual-agent architecture** with order assignment and rider routing coordination, achieving 8–12% delivery time reductions on real Meituan datasets. The second work Wang et al. [2023] extends this to an **online DRL framework** with rider-centered optimization, demonstrating 15% throughput improvements. Jahanshahi et al. Jahanshahi et al. [2022] formulate meal delivery as a **multi-depot vehicle routing problem** with time windows, solving via deep Q-networks (DQN) with prioritized experience replay.

**Applicability to UAM.** These food delivery papers Wang et al. [2024, 2023], Jahanshahi et al. [2022] address **ground vehicle routing** with road network constraints, customer time windows, and rider capacity limits. While conceptually related to UAM logistics, the problem structures differ fundamentally: (1) Ground delivery has **spatial routing** on 2D networks vs our **vertical queueing** through altitude layers; (2) Food delivery optimizes **discrete assignment decisions** vs our **continuous service rate control** + discrete emergency transfers; (3) Delivery systems have **hard time windows** vs our **soft delay penalties** within queueing framework. We cite these works to acknowledge related DRL applications in urban logistics but emphasize the distinct queueing-theoretic foundations of UAM airspace management.

## 2.7 Research Gaps and Our Contributions

Through systematic review of 28 papers across five research threads, we identify **three critical research gaps**:

**Gap 1: Vertical Queueing Theory.** Classical queueing networks Amjath et al. [2024], Efrosinin et al. [2023] assume **horizontal topologies** (manufacturing lines, data centers, transportation networks) without **altitude-dependent dynamics**. UAM airspace exhibits unique vertical constraints where traffic patterns and physical limitations vary by altitude. No existing queueing theory addresses **traffic-aware capacity allocation** or **pressure-triggered cross-layer transfers** under extreme load.

**Gap 2: Capacity Paradoxes and Non-Monotonicity.** Traditional queueing optimization assumes **monotonic capacity-performance relationships**—more servers/buffers yield better throughput/delay Amjath et al. [2024]. UAM systems under 10× high-load violate this assumption: our experiments reveal capacity 10 achieves optimal reward 11,180, while capacity 30 triggers 100% crashes (reward drops to 13, representing −99.8% decline). This **capacity paradox** stems from state space explosion (capacity 10 has $\approx 3^{10}$=59K states vs capacity 23's $\approx 3^{23}$=94B states, differing 1,594,323×) overwhelming learning algorithms. No prior work quantifies such **capacity thresholds** or **structural design principles** (normal pyramid [8,6,4,3,2] vs inverted pyramid [2,3,4,6,8], Cohen's d=2.856).

**Gap 3: Algorithm-Capacity Interactions.** DRL algorithm comparisons Fujimoto et al. [2023], Hessel et al. [2018], Espeholt et al. [2018] evaluate performance on **fixed environments** (MuJoCo, Atari) without systematically varying **system capacity** as an experimental factor. Our 7 configurations × 3 algorithms study reveals **capacity-dependent algorithm degradation**: PPO exhibits severe performance drops at capacity 23–25 (crash rates 40–60%), while A2C maintains robustness (crash rates 10–40%) and TD7 achieves zero crashes. This suggests **on-policy algorithms** (PPO) suffer from non-stationarity under extreme capacity stress, while **actor-critic with experience replay** (TD7) and **synchronous updates** (A2C) provide better stability. No existing work documents such **algorithm-capacity interaction effects**.

**Our Contributions.** Addressing these gaps, we propose the **MCRPS/SD/K framework** with five systematic contributions: (1) **Vertical queueing formulation**—5 altitude layers with traffic-aware capacity allocation and altitude-dependent service rates reflecting physical airspace constraints; (2) **Capacity paradox discovery**—empirical evidence that minimal capacity K=10 outperforms larger configurations, with capacity threshold K=25 marking stability boundary; (3) **Structural design principles**—normal pyramid [8,6,4,3,2] (higher capacity at lower altitudes) demonstrates +124% reward and −36pp crash rate vs inverted pyramid [2,3,4,6,8] at equal total capacity 23; (4) **Algorithmic robustness analysis**—A2C achieves substantially lower crash rates than PPO (16.8% vs 38.8%), while TD7 achieves zero crashes with 100% completion rates; (5) **State space complexity analysis**—capacity expansion creates exponential state space growth ($3^{10}$ = 59K vs $3^{23}$ = 94B, factor

1,594,323×), explaining why minimal capacity outperforms despite smaller buffer sizes.

# 3 System Model and Problem Formulation

## 3.1 Scenario Description

We consider an urban drone delivery network operating in **low-altitude corridors** partitioned into $L = 5$ discrete vertical layers at altitudes 20m, 40m, 60m, 80m, and 100m. The system serves multiple service classes (standard/priority/emergency) with temporally correlated arrivals requiring layer-aware routing and capacity-constrained batch service.

**System Parameters and Units.** We adopt discrete-time modeling with time step $\Delta t = 1$ second. Service rates $\mu_\ell$ are measured in **items per second**, corresponding to single-step completion probability $p_\ell = 1 - e^{-\mu_\ell \cdot \Delta t}$ under exponential service time assumptions. Base arrival rates for the three order classes are $\lambda_1 = 0.30$ s$^{-1}$ (standard), $\lambda_2 = 0.15$ s$^{-1}$ (priority), and $\lambda_3 = 0.05$ s$^{-1}$ (emergency), with total arrival rate $\lambda_{\text{total}} = \sum_k \lambda_k = 0.50$ s$^{-1}$. Actual arrival rate for class $k$ at layer $\ell$ is $\lambda_{k,\ell} = \alpha_\ell \cdot \lambda_k$, where $\alpha_\ell$ denotes layer allocation weights representing traffic distribution across altitudes.

**Physical Constraints and Operational Context.** The vertical stratification reflects real-world urban airspace constraints: lower altitudes face building density, tree canopy, and ground-level activity restrictions, limiting available airspace; higher altitudes offer spatial freedom but introduce coordination complexity and weather exposure. Traffic distribution follows observed patterns in urban logistics: higher layers (L4-L5: 80–100m) handle 55% of orders leveraging cruise-phase efficiency near the nominal 120m flight altitude, while ground-proximate layers (L1-L2: 20–40m) serve 25% for direct building access.

## 3.2 Vertical Layered Airspace

**Layer Structure.** Let $\ell \in \{1, 2, 3, 4, 5\}$ index layers from lowest to highest altitude. Each layer $\ell$ is characterized by:

- **Capacity** $C_\ell$: Maximum number of orders that can queue simultaneously
- **Service rate** $\mu_\ell$: Rate at which orders are completed (items/second)
- **Altitude band** $[h_{\min}^{(\ell)}, h_{\max}^{(\ell)}]$: Vertical extent of the layer

We adopt an **inverted pyramid capacity profile** (higher capacity at higher altitudes) as the baseline configuration, contrasted with normal pyramid in our structural comparison experiments (§6.3):

| Layer | Altitude | Capacity $C_\ell$ | Service Rate $\mu_\ell$ |
|---|---|---|---|
| L5 | 100 m | 8 | 1.20 items/s |
| L4 | 80 m | 6 | 1.00 items/s |
| L3 | 60 m | 4 | 0.80 items/s |
| L2 | 40 m | 3 | 0.60 items/s |
| L1 | 20 m | 2 | 0.40 items/s |

**Physical Rationale for Configuration.** This design reflects real urban low-altitude airspace physics and operational strategies:

1. **Altitude-Dependent Capacity**: Near-ground layers (20–40m) face constrained airspace due to buildings, trees, and pedestrian activity, supporting smaller queues. Higher layers (80–100m) offer open space, accommodating larger buffers. This **inverted pyramid** ($C_{20m}=2 < C_{100m}=8$) matches physical constraints.

2. **Cruise-Proximity Service Rates** ($\mu$ increases with altitude): Lightweight delivery drones operate optimally at 120m cruise altitude, making higher layers more efficient. Layer 5 (100m) enjoys fast horizontal maneuvering close to the 120m cruise corridor ($\mu_5 = 1.2$ fastest), while Layer 1 (20m) requires vertical climb/descent overhead ($\mu_1 = 0.4$ slowest). This gradient reflects transition time penalties between delivery endpoints and cruise altitude.

**Traffic Distribution Pattern.** Order arrivals follow altitude allocation weights $\alpha = [0.10, 0.15, 0.20, 0.25, 0.30]$ from L1 to L5, reflecting delivery logistics: 55% of orders route through higher layers (L4-L5) leveraging cruise efficiency, 25% target ground-proximate layers (L1-L2) for direct building access, and 20% use mid-layer (L3). This creates the **structural matching challenge**: low-traffic L1 (10% of arrivals) must operate with minimal capacity ($C_1=2$), while high-traffic L5 (30% of arrivals) has ample capacity ($C_5=8$). Our experiments (§6.3) demonstrate that this traffic-capacity alignment (inverted pyramid: capacity increases with altitude) outperforms mismatched configurations (normal pyramid) by +124% in reward.

## 3.3 MCRPS/SD/K Network Formulation

The acronym **MCRPS/SD/K** encodes three foundational modeling components addressing limitations of classical queueing theory:

- **MC**: **M**ulti-**C**lass orders with three service priorities (standard/priority/emergency) and temporally **C**orrelated arrivals violating independent Poisson assumptions

- **RPS**: **R**efined **P**oisson **S**plitting mechanism for distributing correlated aggregate flows across vertical layers
- **SD**: **S**tate-**D**ependent control adapting service rates and inter-layer transfers based on real-time queue congestion
- **K**: Finite capacity constraints $\sum_k n_{k,\ell}^t \leq C_\ell$ with blocking/overflow dynamics

**State Variables.** At discrete time $t$, the system state is defined by:

$$n_{k,\ell}^t \ : \text{Number of class-}k \text{ orders queued at layer } \ell \tag{1}$$

$$Q_\ell^t = \sum_{k=1}^{3} n_{k,\ell}^t \ : \text{Total queue length at layer } \ell \tag{2}$$

$$U_\ell^t = Q_\ell^t / C_\ell \ : \text{Occupancy rate (utilization)} \tag{3}$$

**Correlated Multi-Class Arrivals.** Unlike classical M/M/c models assuming independent Poisson arrivals, urban delivery orders exhibit temporal correlation (meal delivery surges during lunch/dinner, weather-triggered clustering). We model aggregate arrivals $A_{\text{total}}^t$ as a compound process, then split to layers and classes. For layer $\ell$ and class $k$, the arrival count at time $t$ is:

$$A_{k,\ell}^t \sim \text{Poisson}(\alpha_\ell \cdot \lambda_k \cdot \Delta t) \tag{4}$$

where $\alpha_\ell$ weights satisfy $\sum_{\ell=1}^{5} \alpha_\ell = 1$. Temporal correlation is captured via time-varying base rates $\lambda_k(t)$ following surge patterns, though our high-load experiments (§6) use constant rates scaled by $10\times$ multiplier to stress-test extreme conditions.

**Validation of Poisson Approximation.** Kolmogorov-Smirnov tests on 10,000-step simulation traces confirm layer-wise inter-arrival intervals closely follow exponential distributions (typical $p$-values 0.3–0.8), validating conditional Poisson splitting for moderate correlation (correlation matrix elements 0.1–0.4). The approximation deteriorates under strong correlation (elements $\geq$0.7), where multivariate Poisson or Cox process modeling would be required—deferred to future work.

**Stochastic Batch Service.** Each layer's service capacity $\text{service\_capacity}_\ell = \min(C_\ell, Q_\ell^t)$ determines the maximum orders serviceable per time step. Theoretically, completed orders should follow **binomial batch service**:

$$S_\ell^t \sim \text{Binomial}(\text{service\_capacity}_\ell,\, p_\ell) \quad \text{where} \quad p_\ell = 1 - e^{-\mu_\ell \Delta t} \tag{5}$$

**Current Implementation Note.** For computational efficiency, our experiments use a **Poisson approximation** $S_\ell^t \sim \text{Poisson}(\mu_\ell) + 1$ (+1 to avoid zero-service), which pro-

vides reasonable stochastic modeling for $\mu_\ell \in [0.4, 1.2]$ with stable performance metrics (§6). Full binomial batch service implementation (algorithm in Appendix B.2) is reserved for future work requiring precise service batch distribution modeling. Within-batch processing follows FCFS for same-class orders, with cross-class prioritization (emergency > priority > standard).

**Pressure-Triggered Downward Transfers.** Orders within queues can transfer between layers based on **pressure differentials** and fairness constraints, primarily implementing **downward transfers** (higher to lower layers) to balance loads during congestion surges. Transfer decisions use probabilistic mechanisms driven by **pressure gradients** $\Delta P_\ell^t$ and capacity states (detailed in §4.1).

**Design Rationale for Downward Transfers.** Although higher layers (L5: $C = 8$, $\mu = 1.2$) offer faster service, downward transfers to lower layers serve as **temporary overflow buffers** preventing global blockage during extreme peaks:

- **Normal operation**: System prioritizes high-layer efficiency via dynamic service rate modulation, rarely triggering transfers (experimental frequency <0.2%)
- **Peak shaving**: During traffic bursts, L5→L1 transfers reduce peak queue lengths by 37%, providing emergency relief
- **Physical plausibility**: Lower layers, though slower ($\mu_1 = 0.4$), serve as short-term spillover buffers superior to rejecting orders; backlog gradually clears via priority mechanisms after high-layer pressure subsides

**Stability Sufficient Conditions.** Under arrival control soft-constraining each layer to $\rho_\ell < 1$ (where $\rho_\ell = \lambda_\ell/(\mu_\ell C_\ell)$ is traffic intensity), downward transfers diluting high-layer congestion guarantee **global queue positive recurrence** (Harris recurrence) and **fluid limit convergence** (Kurtz theorem). Detailed drift analysis appears in Appendix C.

**State-Dependent Control.** Control policies depend on **global queue state** $\mathbf{s}_t = \{n_{k,\ell}^t\}$ (§4.2) and enforce **finite capacity** constraints $\sum_k n_{k,\ell}^t \leq C_\ell$ with blocking for overflow attempts.

**Capacity Constraint Enforcement.** When $Q_\ell^t = C_\ell$ (layer full), new arrivals are either:

- **Blocked**: Rejected with penalty (crash event in evaluation metrics)
- **Redirected**: Routed to adjacent layer with available capacity

Our DRL policies (§4.3) learn to balance service rate allocation and preventive transfers to minimize blocking probability, with varying robustness across algorithms under high-load conditions.

## 3.4 Multi-Objective Problem Statement

We formulate the control problem as a **multi-objective Markov decision process** (MOMDP) optimizing five core objectives plus one diagnostic metric. Objectives $J_1$–$J_5$ drive training, while $J_6$ (transfer efficiency) is evaluated post-hoc for diagnostic purposes.

**Six Evaluation Objectives (Fixed Definitions).** All objectives are measured over evaluation episodes of length $T = 200$ steps for on-policy algorithms (A2C, PPO) and $T = 10{,}000$ steps for off-policy algorithms (TD7):

1. **$J_1$ Throughput** (↑): Mean completed orders per step, items/step. Higher is better.

$$J_1(\pi) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\ell=1}^{5} S_\ell^t \tag{6}$$

2. **$J_2$ Average Delay** (↓): Mean waiting time from arrival to completion (time steps). Lower is better. Let $N_c$ = number of completed orders, $t_c$ = completion time, $t_a$ = arrival time.

$$J_2(\pi) = \frac{1}{N_c} \sum_{i=1}^{N_c} (t_{c,i} - t_{a,i}) \tag{7}$$

3. **$J_3$ Fairness (1−G)** (↑): Load balance across layers via Gini coefficient complement. Higher is better (more fair).

$$J_3(\pi) = 1 - G_t \quad \text{where} \quad G_t = \frac{\sum_{i=1}^{5} \sum_{j=1}^{5} |U_i - U_j|}{2 \cdot 5^2 \cdot \bar{U}} \tag{8}$$

with $\bar{U} = \frac{1}{5} \sum_{\ell=1}^{5} U_\ell$ the mean occupancy rate.

4. **$J_4$ Stability** (↑): Inverse of queue length standard deviation (normalized). Higher is better (more stable).

$$J_4(\pi) = 1 - \frac{\sigma_Q}{\sigma_{\max}} \quad \text{where} \quad \sigma_Q = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (Q_{\text{total}}^t - \bar{Q})^2} \tag{9}$$

5. **$J_5$ Safety** (↑): Complement of violation event rate (overflow/blocking). Higher is better (safer).

$$J_5(\pi) = 1 - \frac{N_{\text{violations}}}{T} \tag{10}$$

6. **$J_6$ Transfer Efficiency**: Ratio of inter-layer transfers to total orders, **diagnostic only** (not optimized). Ideal value <0.2% indicates minimal reliance on emergency

transfers.

$$J_6(\pi) = \frac{\sum_{t=1}^{T} \sum_{\ell=1}^{4} \mathbb{1}[\text{transfer}_{\ell \to \ell-1}^{t} > 0]}{\sum_{t=1}^{T} A_{\text{total}}^{t}} \tag{11}$$

**Training Objective Mapping.** Training optimizes $J_1$–$J_5$ via weighted sum (details §4.3), where $J_2$ (delay) is approximated by **congestion penalty** (queue overflow instant feedback) and $J_4$ (stability) by instability penalty. $J_5$ (safety) is implicitly enforced via congestion penalties (overflow = violation). $J_6$ is computed during evaluation as a diagnostic indicator of transfer mechanism dependency. The composite training objective is:

$$\max_{\pi} J(\pi) = \sum_{i=1}^{5} w_i J_i(\pi) \tag{12}$$

subject to capacity constraints $\sum_k n_{k,\ell}^t \leq C_\ell$, flow conservation, and feasibility constraints. Training reward weight design appears in §4.3.

**Pareto Optimality.** A policy $\pi^*$ is **Pareto-optimal** if no alternative policy $\pi'$ exists such that $J_i(\pi') \geq J_i(\pi^*)$ for all $i \in \{1, \ldots, 5\}$ with strict inequality for at least one objective. We identify the Pareto frontier via systematic sampling across algorithm hyperparameters, weight configurations, random seeds, and threshold settings (§4.2), yielding 262 Pareto-optimal solutions from 10,000 evaluated samples (2.62%). Multi-criteria decision making (MCDM) extracts 13 representative knee-points for decision support.

# 4 Framework and Methodology

The MCRPS/SD/K framework integrates queueing theory, pressure-triggered control mechanisms, Pareto multi-objective optimization, and deep reinforcement learning to address the challenges of vertical airspace management under extreme load conditions. Figure 2 illustrates the complete system architecture with five integrated layers. This section presents the three core methodological components: pressure-triggered transfer mechanisms (§4.1), Pareto optimization methodology (§4.2), and deep RL integration with queueing-aware state-action design (§4.3).

## 4.1 Pressure-Triggered Transfer Mechanisms

**Design Rationale for Downward Transfers.** Although higher layers (L5: $C = 8$, $\mu = 1.2$) offer faster service and larger capacity, downward transfers to lower layers serve as **temporary overflow buffers** to prevent global blockage during extreme congestion peaks. The key intuition operates at three levels:

- **Normal operation**: The system prioritizes high-layer efficiency via dynamic service rate modulation ($\mu$ adjustment), rarely triggering transfers (experimental frequency <0.2%)
- **Peak shaving**: During traffic bursts, L5→L1 transfers reduce peak queue lengths by 37%, providing emergency relief
- **Physical plausibility**: Lower layers, though slower ($\mu_1 = 0.4$), serve as short-term spillover buffers superior to order rejection; backlog gradually clears via priority mechanisms after high-layer pressure subsides

**Stability Sufficient Conditions.** Under arrival control (via `arrival_multiplier` action, §4.3) soft-constraining each layer to $\rho_\ell < 1$ (where $\rho_\ell = \lambda_\ell/(\mu_\ell C_\ell)$ is traffic intensity), downward transfers diluting high-layer congestion guarantee **global queue positive recurrence** (Harris recurrence) and **fluid limit convergence** (Kurtz theorem). Detailed drift analysis appears in Appendix C.

**Pressure Metric.** For layer $\ell$ at time $t$, we define a composite pressure metric:

$$P_\ell^t = \beta_1 \left( \frac{Q_\ell^t}{C_\ell} \right) + \beta_2 \left( 1 - \frac{\mu_\ell^t}{\mu_{\max,\ell}} \right) + \beta_3 G_\ell^t \tag{13}$$

where $\beta_1$, $\beta_2$, $\beta_3$ are pressure weight coefficients, $\mu_{\max,\ell}$ denotes the maximum nominal service rate for layer $\ell$, and $G_\ell^t$ represents local fairness measured via Gini coefficient. The pressure differential $\Delta P_\ell^t = P_\ell^t - \bar{P}^t$ (where $\bar{P}^t$ is the mean pressure across all layers) drives transfer decisions.

**Implementation Note.** The complete pressure formula (Equation 13) represents the theoretical framework design for conceptual modeling. **Current experimental implementation uses a simplified version** $P_\ell = Q_\ell/C_\ell$ (i.e., $\beta_1 = 1, \beta_2 = \beta_3 = 0$) for three reasons: (1) **Dominant factor**: Under the inverted pyramid capacity design, the congestion term $Q_\ell/C_\ell$ is the core driver for triggering transfers (capacity varies significantly from $C_1 = 2$ to $C_5 = 8$); (2) **Service rate encoded in DRL state**: The service redundancy term $(1 - \mu_\ell^t/\mu_{\max,\ell})$ is already exposed to the RL agent through the `service_rates` dimension in the state space, avoiding redundant encoding in the pressure metric; (3) **Fairness optimized via reward function**: Local fairness $G_\ell^t$ is optimized through the global Gini coefficient reward term (weight +5, see §4.3). The simplified version demonstrates stable performance in experiments; full formula implementation with $\beta$ weight tuning is reserved for future work exploring more complex pressure modeling strategies.

**Transfer Strategy (Current Implementation).**

- **Primary transfer direction**: When source layer pressure significantly exceeds target layer pressure ($\Delta P > \theta_{\text{transfer}}$, where $\theta_{\text{transfer}} = 0.3$ calibrated via grid search over $\{0.1, 0.2, 0.3, 0.4\}$ achieving optimal fairness-throughput tradeoff) and target layer has spare capacity, execute **downward transfers** (L5→L4→ $\cdots$ →L1)

- **Transfer triggering conditions**: (1) Order waiting time exceeds threshold; (2) Source-target pressure differential exceeds threshold; (3) Target layer has not reached capacity limit; (4) Global fairness constraint satisfied

- **Load balancing mechanism**: Transfers from high layers (large $C$, large $\mu$) to low layers (small $C$, small $\mu$) primarily serve emergency regulation rather than routine operation. System design prioritizes high-layer efficiency via **service rate modulation** ($\mu$ dynamic adjustment) over frequent transfers

- **Stability guarantee**: Transfers activate only when pressure differentials are significant, avoiding unnecessary inter-layer perturbations

**Fairness Control via Gini Coefficient.** We monitor global fairness through the **Gini index** over effective layer workloads $x_i = U_i = Q_i/C_i$ (occupancy rates at each layer):

$$G_t = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i - x_j|}{2n^2 \bar{x}} \tag{14}$$

where $n = 5$ is the number of layers and $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ is the arithmetic mean of occupancy rates. A soft constraint $G_t \leq G_{\text{target}} + \epsilon$ regulates transfers, where $G_{\text{target}}$ is the target upper bound and $\epsilon$ is a relaxation term.

**Note on Fairness Metric Direction.** In the reward function and evaluation, we use **Fairness = 1 - $\mathbf{G}_t$** as the optimization objective (direction ↑), making the fairness metric directionally consistent with other performance metrics (higher is better). The Gini coefficient $G_t$ itself decreases as fairness improves, while $(1 - G_t)$ increases.

## 4.2 Pareto Multi-Objective Optimization Methodology

**Pareto Optimality Definition with Unified Objective Directions.** To unify comparison criteria, **Pareto and hypervolume (HV) computation convert all objectives to 'higher-is-better' utility form**: objectives requiring minimization are negated ($\mathbf{J}'_2 = -\mathbf{Delay}$; if including transfer dependency, $\mathbf{J}'_6 = \mathbf{1} - \mathbf{TransferRate}$), while remaining objectives ($J_1$ Throughput↑, $J_3$ Fairness↑, $J_4$ Stability↑, $J_5$ Safety↑) are already in "higher-is-better" form and retain original direction. Formally, for the six-dimensional objective vector $\mathbf{f}(\mathbf{x}) = [J_1, -J_2, J_3, J_4, J_5, 1 - J_6]$ (all maximized), solution $\mathbf{x}^*$ is Pareto-optimal if and only if

there does not exist another solution $\mathbf{x}$ such that $f_i(\mathbf{x}) \geq f_i(\mathbf{x}^*)$ for all $i$ with strict inequality for at least one $j$.

**Pareto Frontier Identification.** We employ the efficient non-dominated sorting algorithm to identify the Pareto-optimal solution set. For each solution, we compute its domination count; solutions with zero domination count constitute the Pareto frontier. To ensure solution feasibility, we introduce a stability threshold filter (Stability $\geq 0.5$) to exclude system crash solutions.

**Hypervolume Computation.** Hypervolume (HV) measures Pareto frontier quality, defined as HV = Volume($\{\mathbf{y} \mid \exists \mathbf{x} \in \text{PF} : \mathbf{f}(\mathbf{x}) \prec \mathbf{y} \prec \mathbf{r}\}$), where $\mathbf{r}$ is the reference point. **Normalization method**: Each objective is normalized to the $[0, 1]$ interval using min-max scaling $f_i' = (f_i - f_{i,\min})/(f_{i,\max} - f_{i,\min})$, where $f_{i,\min}/f_{i,\max}$ are the minimum/maximum values across all 10,000 evaluated solutions for objective $i$. **Reference point selection**: $\mathbf{r} = [0, 0, 0, 0, 0, 0]$ (worst value for each objective after normalization), ensuring HV computation consistency and comparability. Figure captions in §6.5 adopt this unified specification.

**Knee-Point Detection Method.** Knee-points are the most representative solutions on the Pareto frontier, representing critical objective tradeoff inflection points. We employ a multi-criteria decision making (MCDM) approach to select knee-points:

1. **Quality criterion (40%)**: Compute each Pareto solution's normalized distance to the ideal point $d_{\text{ideal}} = \|\mathbf{f}(\mathbf{x}) - \mathbf{f}^*\|$, where $\mathbf{f}^* = [1, 1, 1, 1, 1, 1]$ represents the best achievable value for each normalized objective (all objectives scaled to $[0, 1]$ via min-max normalization); smaller distance indicates higher quality

2. **Diversity criterion (40%)**: Sparsity score based on K-nearest neighbor distance $s_{\text{div}} = \text{mean}(\text{dist}_{k\text{-nearest}})$; larger distance indicates stronger representativeness

3. **Balance criterion (20%)**: Objective balance based on coefficient of variation $CV = \sigma/\mu$; smaller CV indicates better balance across objectives

The composite score is Score $= 0.4 \cdot q_{\text{quality}} + 0.4 \cdot q_{\text{diversity}} + 0.2 \cdot q_{\text{balance}}$. We select the top $K = \max(5, \min(15, \text{round}(|\text{PF}|/20)))$ solutions as knee-points (using rounding, e.g., for $|\text{PF}| = 262$, we get $262/20 \approx 13$), where $|\text{PF}|$ is the Pareto frontier size.

## 4.3 Deep Reinforcement Learning Integration

**State Space (29-Dimensional Dict).** We adopt a compact, efficient state design comprising 7 key observation dimensions:

1. **Queue lengths** (`queue_lengths`, 5-dim): Current queue length at each layer

2. **Occupancy rates** (`occupancy_rates`, 5-dim; code: `utilization_rates`): Normalized ratio $U_\ell = Q_\ell/C_\ell$ representing occupancy/congestion, *not* traffic intensity $\rho = \lambda/(\mu \cdot C)$

3. **Queue change rates** (`queue_changes`, 5-dim): Queue length changes between adjacent time steps

4. **Load rates** (`load_rates`, 5-dim): System load indicator as arrival rate over service capacity $\rho_\ell = \lambda_\ell/(\mu_\ell \cdot C_\ell)$ (**Note**: This is the true traffic intensity in queueing theory; stability requires $\rho < 1$)

5. **Service rates** (`service_rates`, 5-dim): Current actual service rate at each layer

6. **Previous reward** (`prev_reward`, 1-dim): Reward feedback from the previous step

7. **System metrics** (`system_metrics`, 3-dim): [Overall load, average occupancy rate, stability score]

This design avoids redundant information and focuses on core features of queue dynamics. **Terminology clarification**: Occupancy rate $U = Q/C$ measures capacity utilization (0-1 indicates normal, >1 indicates overflow); traffic intensity $\rho = \lambda/(\mu \cdot C)$ measures system load strength (<1 is a necessary condition for stability).

**Action Space (11-Dimensional Dict, Mixed Discrete-Continuous).**

1. **Continuous actions (6-dim)**:

   - `service_intensities` (5-dim): Controls service intensity multiplier for each layer $\in [0.1, 2.0]$
   - `arrival_multiplier` (1-dim, implementation variable name): Adjusts global arrival rate multiplier $\in [0.5, 5.0]$

   **Semantic constraint on arrival_multiplier**: This parameter represents **system-side throttling/gating** (e.g., queue entrance quota or temporary holding mechanism), belonging to control-layer strategy rather than demand-side natural arrivals. When enabled, actual arrival rate becomes $\lambda'_k = m \cdot \lambda_k$ (where $m$ is the policy-output multiplier), used for emergency peak-shaving or capacity protection.

   **Stability safety gate (explicit projection)**: During execution, we apply **soft clipping** to `arrival_multiplier` ensuring all layers satisfy $\rho_\ell < 1$. The projection formula is:

$$m^* = \min\left(m_{\text{proposed}}, \max\{m : \max_\ell \rho_\ell(m) < 1\}\right) \tag{15}$$

where $\rho_\ell(m) = (m \cdot \lambda_\ell)/(\mu_\ell \cdot C_\ell)$.

2. **Discrete actions (5-dim)**: `emergency_transfers` (MultiBinary×5), emergency transfer flag for each layer; 1 indicates trigger downward transfer to the layer below (i.e., layer $\ell \to$ layer $\ell - 1$), 0 indicates maintain current state

**Reward Function Design and Training-Evaluation Objective Mapping.** We employ a multi-objective weighted-sum reward structure comprising positive incentives ($J_1$ throughput, $J_3$ fairness, energy efficiency $E$) and penalty terms (congestion proxy for $J_2$, instability proxy for $J_4$). Training optimizes 5 core components, where $\mathbf{J}_2$ **delay** is approximated by **congestion penalty** (instantaneous negative feedback for queue overflow, providing faster signal than delay measurement), $\mathbf{J}_4$ **stability** via instability penalty, $\mathbf{J}_5$ **safety** implicitly enforced through congestion penalties (overflow = violation), and $\mathbf{J}_6$ **transfer efficiency** computed only during evaluation as a diagnostic metric.

**Energy efficiency $E$ definition**: Normalized indicator taking values in $[0, 1]$ (higher = better):

$$E = \frac{\text{completed orders/total energy}}{E_{\max}}$$

Total energy follows a simplified model: higher service rate $\mu$ yields lower per-unit cost; altitude closer to 120m cruise layer has lower energy cost.

The composite training reward is:

$$R_t = 10 \cdot J_1 + 5 \cdot J_3 + 3 \cdot E - 20 \cdot \text{Congestion} - 15 \cdot \text{Instability} + 2 \cdot \text{Transfer}_{\text{benefit}} + 2 \cdot \text{Stability}_{\text{bonus}} \tag{16}$$

**Weight design rationale**:

1. **Congestion penalty weight ($-20$) exceeds throughput reward ($+10$)**: Based on queueing theory stability constraint $\rho < 1$, the system must prioritize avoiding queue overflow to maintain long-term stability. Congestion leads to cascading failures with costs far exceeding single-step throughput loss, hence the 2:1 penalty-to-reward ratio.

2. **Fairness weight ($+5$) is 50% of throughput**: Reflects system emphasis on load balancing ($J_3$), preventing individual layer overload while others remain idle, embodying urban airspace "safety > performance > cost" priority.

3. **Instability penalty ($-15$) slightly lower than congestion penalty**: Queue oscillations affect service quality but do not immediately cause system collapse, therefore penalty strength is set at 75% of congestion penalty.

4. **Energy efficiency weight (+3) is lowest**: Energy efficiency ($E$) as a long-term optimization goal has lower immediate reward value than throughput and fairness, set at 30% of baseline throughput weight. **Energy efficiency ($E$) $\neq$ transfer efficiency**: Training reward's energy efficiency ($E$) measures energy consumption efficiency, completely independent from evaluation metric $J_6$ transfer efficiency (diagnosing inter-layer transfer dependency).

5. **Transfer benefit reward (+2)**: When transfers successfully alleviate high-layer pressure, reward is granted to encourage effective emergency regulation. Calculation: when `upper_pressure` > `lower_util`, $\text{transfer}_{\text{benefit}} = +2.0 \times$ transfer count.

6. **System stability reward (+2)**: Stability reward based on queue changes over the most recent 10 steps, $\text{stability}_{\text{bonus}} = +2.0 \times \exp(-\text{avg\_change}/2.0)$, encouraging smooth operation.

**Training reward weights**: [Throughput:Fairness:Energy:Transfer:Stability:Congestion:Instability] = [10:5:3:2:2:−20:−15], where the first 5 items are positive incentives and the last 2 are penalties. Transfer and stability are auxiliary reward terms; primary weights remain the first 3 items (throughput/fairness/energy). Sensitivity analysis validates algorithm ranking remains stable under ±20% perturbation range.

**Evaluation reports complete 6 objectives**: $\{J_1$ Throughput ↑, $J_2$ Delay ↓, $J_3$ Fairness ↑ , $J_4$ Stability ↑, $J_5$ Safety ↑, $J_6$ Transfer Efficiency (diagnostic)$\}$, where $J_2/J_4$ use actual measured values, and $J_6$ diagnoses whether the system over-relies on transfers (ideal <0.2%, primarily relying on service rate modulation).

**Algorithm Adapters.** Discrete RL algorithms (Rainbow, R2D2) use adaptive discretization + prioritized experience replay (PER); continuous RL algorithms (PPO, TD7, SAC-v2, TD3, DDPG) employ separate policy/value heads for mixed actions; distributed RL (IMPALA) applies V-trace with queueing-aware batch processing.

# 5 Experimental Setup

## 5.1 Environment Configuration and Operational Parameters

**Simulation Platform.** We implement the vertical queueing system using a custom OpenAI Gym environment `DRLOptimizedQueueEnvFixed` with deterministic initialization (`seed=42`) to ensure reproducibility. The environment models discrete-time dynamics with $\Delta t = 1$ second time steps, simulating 5 vertical layers at altitudes 20m, 40m, 60m, 80m, and 100m.

**Airspace Parameters.** The baseline configuration adopts an **inverted pyramid** capacity profile reflecting real-world low-altitude constraints:

| Layer | Altitude | Capacity $C_\ell$ | Service Rate $\mu_\ell$ | Arrival Weight $\alpha_\ell$ |
|---|---|---|---|---|
| L5 | 100 m | 8 | 1.20 items/s | 0.30 |
| L4 | 80 m | 6 | 1.00 items/s | 0.25 |
| L3 | 60 m | 4 | 0.80 items/s | 0.20 |
| L2 | 40 m | 3 | 0.60 items/s | 0.15 |
| L1 | 20 m | 2 | 0.40 items/s | 0.10 |

Total system capacity $C_{\text{total}} = \sum_{\ell=1}^{5} C_\ell = 23$. Traffic allocation weights $\alpha = [0.10, 0.15, 0.20, 0.25, 0.30]$ represent realistic delivery patterns where 55% of orders route through upper layers (L4-L5) leveraging cruise efficiency.

**Physical Basis for Service Rate Gradient.** The monotonically increasing service rates $\mu \in [0.4, 1.2]$ (L1→L5) reflect the operational physics of lightweight delivery drones (e.g., DJI Matrice series, payload ≤5kg) in urban airspace. The **120m altitude** serves as the primary cruise layer for horizontal transit due to: (1) regulatory clearance above building clutter (typical urban high-rise ≈60–80m); (2) optimal lift-to-drag ratio for quadcopter aerodynamics; (3) reduced ground-level obstacle avoidance overhead. Consequently, higher layers (L4-L5: 80–100m) operate closer to cruise conditions with minimal vertical maneuvering, achieving faster order processing ($\mu_5 = 1.2$). Conversely, lower layers (L1-L2: 20–40m) require frequent climb/descent transitions between delivery points and cruise altitude, incurring time penalties that reduce effective service rates ($\mu_1 = 0.4$). This 3:1 service rate gradient ($\mu_5/\mu_1 = 3.0$) aligns with field measurements showing 2.5–3.5× throughput differences between ground-proximate and cruise-proximate operations.

**Order Categories and Service Priorities.** The system handles three order classes with differentiated service priorities:

- **Standard orders** (60% of arrivals, $\lambda_1 = 0.30$ s$^{-1}$): Regular delivery requests with relaxed time constraints
- **Priority orders** (30% of arrivals, $\lambda_2 = 0.15$ s$^{-1}$): Time-sensitive deliveries (e.g., meal delivery within 30 minutes)
- **Emergency orders** (10% of arrivals, $\lambda_3 = 0.05$ s$^{-1}$): Critical logistics (e.g., medical supplies, urgent documents)

Base arrival rate $\lambda_{\text{total}} = \sum_k \lambda_k = 0.50\ \text{s}^{-1}$. Under 10× high-load stress testing, the system operates at $\lambda'_{\text{total}} = 5.0\ \text{s}^{-1}$, corresponding to average system utilization:

$$\bar{\rho} = \frac{\sum_{\ell=1}^{5} \alpha_\ell \lambda'_{\text{total}}}{\sum_{\ell=1}^{5} \mu_\ell C_\ell} = \frac{5.0}{1.84 \times 23} \approx 1.84 \quad (184\%\ \text{load}) \tag{17}$$

**Real-World Motivation for Extreme Load.** The 184% load regime reflects realistic surge scenarios in UAM operations: (1) **Meal delivery peaks**: Urban food delivery services experience 5–10× demand spikes during lunch (11:30–13:00) and dinner (18:00–20:00) rushes, as documented in logistics optimization literature where peak-to-average ratios reach 8–12× in dense metropolitan areas; (2) **Emergency evacuations**: Natural disasters or public health emergencies trigger concentrated medical/supply delivery requests exceeding nominal capacity; (3) **Event-driven surges**: Major sporting events or festivals create localized hotspots with >10× baseline traffic. While $\rho > 1$ violates steady-state queueing theory assumptions, modern urban logistics increasingly operate in **transient overload regimes**, where systems must maintain acceptable service during temporary capacity violations through dynamic control (rate modulation, cross-layer transfers) rather than static provisioning. Our 10× stress test evaluates algorithm robustness under these critical—yet operationally realistic—conditions.

**Thresholds and Safety Constraints.** System monitoring employs: occupancy safety threshold $U_{\text{safe}} = 0.85$; Gini fairness target $G_{\text{target}} = 0.30$; queue overflow limit $Q_{\max} = 10$ (per-layer emergency threshold).

## 5.2 Baseline Methods and Algorithms

We evaluate **15 methods** (10 reinforcement learning algorithms + 5 traditional schedulers) across 7 capacity configurations. A random policy serves as stress-test baseline but is excluded from the formal 15-method comparison.

### 5.2.1 Reinforcement Learning Methods (10 Algorithms)

**1. A2C (Advantage Actor-Critic) with Staged Learning Rate Scheduling.** We employ a two-stage learning rate schedule optimized for the vertical queueing domain:

- **Exploration phase** (0–200k steps): Fixed high learning rate $\eta = 7 \times 10^{-4}$ to enable aggressive policy space exploration
- **Convergence phase** (200k–500k steps): Annealing from $7 \times 10^{-4} \rightarrow 1 \times 10^{-5}$ for fine-grained policy refinement

The transition point (≈200k steps) corresponds to initial policy stabilization, after which reduced learning rates minimize oscillations and enable superior final performance. This staged approach avoids premature annealing (causing exploration insufficiency) and fixed-rate plateaus (limiting convergence quality). Network architecture: [512, 512, 256] (enhanced capacity to capture 5-layer structural dependencies). Hyperparameters: $n_{\text{steps}} = 32$, GAE $\lambda = 0.95$, entropy coefficient 0.01.

**2. PPO (Proximal Policy Optimization).** Standard implementation with clipping parameter $\epsilon = 0.2$, GAE $\lambda = 0.95$, network [256, 256]. Learning rate $3 \times 10^{-4}$ with standard cosine annealing.

**3. TD7 (TD3 + SALE + Adaptive Exploration + LAP + Checkpoints).** TD7 extends TD3 with five synergistic components detailed in Table 2. The algorithm demonstrates unique dual-jump learning dynamics (§6.4).

**Synergistic Effects.** The combination of SALE + LAP + Checkpoints produces the dual-jump learning phenomenon: Jump 1 (+857% at 26,689 steps) marks SALE representation breakthrough, Jump 2 (+95% at 26,989 steps, interval ≈300 steps) corresponds to policy optimization convergence. This tight coupling (completion within <1,500 steps) reveals strong component synergy.

**4–10. Additional RL Methods.** R2D2 (LSTM memory for partial observability, hidden state storage, $n$-step=5); Rainbow DQN (51-atom distributional, $n$-step=3, stability-optimized); SAC-v2 (adaptive temperature); SAC (standard version without temperature adaptation); TD3 (baseline for TD7 comparison); DDPG (baseline for algorithm evolution study); IMPALA (conservative V-trace with $\bar{\rho} = 1.0$, $\bar{c} = 1.0$).

### 5.2.2 Traditional Scheduling Baselines (5 Methods)

**Heuristic Scheduler (Domain-Tuned).** Hand-crafted rules leveraging domain knowledge: prioritize emergency orders, balance layer loads via heuristic pressure thresholds, employ greedy service rate allocation. Serves as upper bound for non-learning approaches.

**Other Schedulers.** Proportional fair scheduling (equalizes service shares), priority-based (strict class ordering), FCFS (first-come-first-served), SJF (shortest-job-first based on estimated service times). These classical methods provide lower-bound baselines.

**Random Policy.** Uniformly random action selection; used solely for stress-testing environment stability, excluded from the 15-method formal comparison.

### 5.2.3 Unified Hyperparameter Panel

Table 3 summarizes training hyperparameters with method-specific configurations highlighted.

**A2C Staged Learning Rate Timeline:**

| Phase | Steps | LR | Performance |
|---|---|---|---|
| Exploration | 0–200k | $7 \times 10^{-4}$ | $0 \rightarrow 3500$ (4.5 min) |
| Transition | 200k | — | Stability checkpoint |
| Convergence | 200k–500k | $\rightarrow 1 \times 10^{-5}$ | $3500 \rightarrow 4438$ (2.4 min) |
| **Total** | **500k** | **Staged** | **SOTA in 6.9 min** |

**Key Insight.** A2C's staged schedule avoids premature annealing (causing exploration insufficiency, cf. standard decay) while leveraging late-stage fine-tuning to surpass fixed-rate performance ceilings. The transition timing (≈200k steps) corresponds to initial policy stabilization, where reduced learning rates minimize oscillations and boost final performance.

## 5.3 Parameter Selection Rationale

This subsection justifies key experimental design choices based on theoretical grounding, pilot studies, and operational considerations.

**Sample Size (n=3).** The choice of three independent training runs per configuration balances statistical validity with computational constraints. While larger sample sizes would improve power for detecting small effect sizes, our structural comparison experiments yielded **extremely large effect sizes** (Cohen's $d = 33.6$ for A2C, $d = 273.6$ for PPO) with high statistical significance ($p < 0.001$), indicating that even minimal sample sizes suffice to detect the substantial performance differences observed. The combination of n=3 with extreme effect sizes ensures adequate statistical power (power $> 0.99$ for detecting differences $> 5\%$ in mean rewards). Each independent run employs different random seeds (42, 123, 456) to ensure reproducibility while capturing training variance across initialization conditions. Welch's $t$-test accounts for potential variance heterogeneity between structural configurations, providing robust inference without equal-variance assumptions.

**Traffic Distribution ($\alpha = [0.10, 0.15, 0.20, 0.25, 0.30]$).** The baseline traffic allocation reflects realistic UAM delivery scenarios where upper altitude layers handle disproportionate traffic volumes. This distribution is motivated by operational physics: (1) **Cruise efficiency**—higher layers (80–100m altitude) operate closer to optimal cruise conditions for lightweight delivery drones, minimizing climb/descent overhead and thus attracting long-distance deliveries; (2) **Building clearance**—lower layers (20–40m) face greater ground-

level obstacle density, making them suitable primarily for final approach/departure segments rather than sustained horizontal transit; (3) **Regulatory precedent**—emerging UTM frameworks (NASA UTM, European U-space) designate upper low-altitude airspace (80–120m) as primary corridors for autonomous operations. The 55% upper-layer traffic concentration ($\alpha_4 + \alpha_5 = 0.55$) aligns with drone delivery throughput studies showing 2–3× higher sustained rates at cruise altitudes. We acknowledge this is a **simplified baseline assumption**; Section 7.3 discusses generalizability limitations, and planned sensitivity analyses will validate findings under reversed and uniform traffic patterns.

**Load Configuration (5× Baseline).** The 5× load factor yields average system utilization $\rho \approx 95\%$, positioning the system near queueing-theoretic stability boundaries ($\rho \to 1$) where control policies face maximal challenge. This choice balances two competing requirements: (1) **Stress testing**—loads must be high enough to differentiate effective vs ineffective policies, as low-utilization regimes ($\rho < 0.5$) allow even naive strategies to succeed; (2) **Training stability**—excessively high loads ($\geq 10\times$, $\rho > 1.8$) cause immediate crash-reset cycles that prevent DRL policy convergence (verified via pilot experiments showing 100% crash rates for pyramid structures at 10× load). The 5× level represents realistic peak-demand scenarios (e.g., lunch/dinner delivery rushes with 5–10× baseline traffic surges documented in urban logistics literature), while maintaining 0% crash rates that enable stable policy learning.

**Capacity Configurations.** The seven evaluated configurations (K=10, 20, 23, 25, 30, 40, and structural variants [8,6,4,3,2] vs [2,3,4,6,8]) span a range designed to test the capacity-performance relationship: (1) Minimal configurations (K=10) test lower bounds; (2) Moderate configurations (K=20–25) explore intermediate regimes; (3) High configurations (K=30–40) probe potential upper bounds; (4) Structural variants with identical total capacity (K=23) isolate traffic-capacity matching effects. The inverted pyramid [8,6,4,3,2] baseline aligns with the $\alpha$ distribution (matching high capacity to high traffic layers), while the normal pyramid [2,3,4,6,8] serves as a controlled counterfactual.

**Random Seed Selection.** Seeds [42, 123, 456] provide: (1) Reproducibility (seed 42 is a widely-adopted convention in machine learning research); (2) Independence (non-sequential values reduce risk of correlated initialization artifacts); (3) Practical traceability (simple memorable values facilitate experimental tracking and debugging).

## 5.4 Training Protocol

**Sample Size and Load Configuration.** We conducted **n=3 independent training runs** with different random seeds (42, 123, 456) for each structural configuration (inverted

pyramid [8,6,4,3,2] vs normal pyramid [2,3,4,6,8]). Training was performed under **5× baseline load** (average utilization $\rho \approx 95\%$), which provides sufficient challenge while maintaining training stability (0% crash rate across all configurations). This load level was selected after preliminary experiments at 10× load resulted in 100% crash rates for pyramid structures, preventing meaningful policy learning due to excessive bottom-layer overload ($\rho = 345\%$ at Layer 1).

**Training Configuration.** All reinforcement learning methods train for 500,000 environment steps with evaluation every 5,000 steps. Each evaluation runs 10 episodes (episodes completed to terminal states or timeout). Single-episode step limit: 1,000 steps for training rollouts.

**Hardware Environment.** Experiments execute on Intel Xeon Gold 6248R CPUs (2.5–3.9 GHz, 20 cores), 128 GB DDR4 RAM, NVIDIA A100 40GB GPUs, and NVMe SSDs for experience replay storage. Average training wall-clock time: 6.9 minutes per 500k steps (A2C with staged scheduling).

**Evaluation Protocol and Critical Distinction. Important caveat**: Different algorithm families employ different evaluation episode lengths, rendering raw reward values **non-comparable**:

- **On-policy algorithms** (A2C, PPO): `max_episode_steps=200`, reflecting standard on-policy evaluation protocols
- **Off-policy algorithms** (TD7, SAC, TD3, DDPG, etc.): `max_episode_steps=10,000`, enabling long-horizon value estimation

Consequently, TD7's average reward (375,294) vastly exceeds A2C/PPO (≈4,400–6,500) purely due to episode length (10,000 vs 200 steps), **not algorithmic superiority**. Direct reward comparison is invalid. Instead, we compare **robustness metrics** (crash rates, completion rates, stability) which are normalized and length-independent. This protocol distinction must be emphasized to avoid misleading interpretations.

## 5.5 Performance Metrics and Statistical Analysis

**Performance Indicators.** We measure six core objectives defined in §3:

1. $\mathbf{J}_1$ **Throughput** (items/step↑): Mean completed orders per time step
2. $\mathbf{J}_2$ **Average Delay** (steps↓): Mean waiting time from arrival to completion
3. $\mathbf{J}_3$ **Fairness** (1−G↑): Gini coefficient complement over layer occupancy rates

4. $\mathbf{J}_4$ **Stability** (↑): Normalized inverse of queue length standard deviation, defined as Stability $= 1 - \sigma_Q/\sigma_{\max}$ where $\sigma_Q = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(Q_{\text{total}}^t - \bar{Q})^2}$ is the steady-state queue length standard deviation. **Normalization calibration**: $\sigma_{\max}$ is set to the maximum $\sigma_Q$ observed across all 15 methods in all experimental replications; empirically $\sigma_{\max} = 472.3$, ensuring Stability $\in [0, 1]$. **Stability threshold**: Stability $\geq 0.5$ indicates $\sigma_Q \leq 0.5 \cdot \sigma_{\max} = 236.2$; used to filter Pareto frontiers, excluding oscillatory crash solutions (details in Supplementary Table S3).

5. $\mathbf{J}_5$ **Safety** (↑): Complement of violation event rate. **Violation definition**: A violation occurs when (1) $Q_\ell > C_\ell \times 1.2$ (20% overflow) or (2) any layer experiences continuous overload ($Q_\ell > C_\ell$) for >10 consecutive steps. Safety $= 1-(N_{\text{violations}}/T)$, where higher values indicate safer operation.

6. $\mathbf{J}_6$ **Transfer Efficiency** (diagnostic): Ratio of inter-layer transfer events to total arrivals; ideal value <0.2% indicates minimal reliance on emergency transfers

**Learning Metrics.** Sample efficiency (area under learning curve), convergence speed (steps to 90% final performance), final performance plateau, training variance, wall-clock time.

**Statistical Significance Testing.** For 15 methods (10 RL + 5 schedulers), we perform pairwise comparisons totaling $\binom{15}{2} = 105$ tests. We employ **Welch's $t$-test** (does not assume equal variances) to assess mean differences, computing **Cohen's** $d$ effect size ($|d| \geq 0.8$: large effect; $0.5 \leq |d| < 0.8$: medium; $0.2 \leq |d| < 0.5$: small) and **95% confidence intervals (CI)**. Due to multiple comparisons, we apply **Bonferroni correction** to control family-wise error rate (FWER): adjusted significance level $\alpha' = \alpha/105 = 0.05/105 \approx 0.000476$. We reject the null hypothesis only when $p_{\text{adj}} < \alpha'$, ensuring overall Type I error rate $\leq 5\%$. For non-normally distributed data (Shapiro-Wilk test $p < 0.05$), we substitute **Mann-Whitney U test**, applying identical Bonferroni correction.

**Statistical Reporting Convention. Unless otherwise noted, all significance results in this paper report Bonferroni-corrected $p$-values (denoted $p_{\text{adj}}$)** with family-wise error control at $\alpha' = 0.000476$. Interval estimates provide 95% CI and Cohen's $d$. Complete statistical details for key pairwise comparisons (e.g., A2C vs PPO, TD7 vs TD3) appear in Supplementary Table S1. Kolmogorov-Smirnov tests for arrival processes report **single-test $p$-values** (not subject to multiple comparison correction).

# 6 Results and Analysis

We present comprehensive results from systematic experiments with 7 capacity configurations × 3 DRL algorithms (A2C, PPO, TD7) × 5 independent seeds, totaling 21 experimental conditions with 100,000 training steps each. All comparisons employ **Bonferroni-corrected significance testing** ($\alpha' = 0.000476$ for 105 pairwise comparisons among 15 methods) as specified in §5. Unless noted otherwise, reported $p$-values are adjusted ($p_{\text{adj}}$).

## 6.1 Capacity Paradox: Minimal Capacity Achieves Optimal Performance

**Counterintuitive Discovery.** Classical queueing theory and intuitive design wisdom suggest that larger buffer capacities yield better performance by accommodating traffic bursts. Our experiments reveal a striking **capacity paradox**: the minimal capacity configuration (K=10, uniform [2,2,2,2,2]) achieves **optimal performance** (average reward 11,180, 0% crash rate), significantly outperforming larger configurations including the baseline inverted pyramid (K=23: reward 8,844, 29% crash) and uniform-25 (K=25: reward 7,817, 35% crash).

Table 4 summarizes performance across 7 capacity configurations, ranked by average reward under A2C and PPO algorithms.

**Critical Findings:**

1. **Capacity Threshold**: Configurations with K $\leq$ 25 maintain system viability (crash rates 0–35%), while K $\geq$ 30 triggers immediate collapse (100% crash, episode length = 1 step).

2. **Performance Cliff**: Capacity increase from 25 $\rightarrow$ 30 causes performance drop from 7,817 $\rightarrow$ 13, representing **99.8% degradation**. This sharp boundary defines a critical stability threshold.

3. **Non-Monotonic Relationship**: Performance exhibits inverted-U shape with respect to capacity, peaking at K=10 then declining as capacity increases, contradicting monotonic scaling assumptions in traditional queueing optimization.

**State Space Complexity Hypothesis.** We hypothesize this paradox stems from exponential state space growth overwhelming learning algorithms. For a 5-layer system with maximum queue length $Q_{\max,\ell} \approx C_\ell$ per layer, the approximate state space size is:

$$|\mathcal{S}| \approx \prod_{\ell=1}^{5} (C_\ell + 1) \approx 3^K \quad \text{(simplified estimate)} \tag{18}$$

Under this approximation:

- **Capacity 10**: $|\mathcal{S}| \approx 3^{10} = 59,049$ states
- **Capacity 23**: $|\mathcal{S}| \approx 3^{23} = 94,143,178,827$ states
- **State space ratio**: 94B / 59K $\approx$ **1,594,323**×

This exponential explosion creates severe sample complexity challenges. With fixed training budget (100,000 steps), algorithms can adequately explore small state spaces (K=10) but suffer from **sparse sampling** in vast spaces (K≥23), leading to suboptimal policies and instability. Capacity ≥30 exacerbates this to the point of immediate training collapse.

**Practical Implications.** The capacity paradox reveals that **adding buffers is not a panacea** for high-load systems. System designers must balance capacity expansion against learning difficulty, particularly in deep RL contexts where state space dimensionality directly impacts sample efficiency. For urban airspace management, this suggests conservative capacity allocation (K=10–20 range) paired with sophisticated control policies may outperform aggressive capacity expansion.

**Extended Training Validation.** To rule out sample complexity as the root cause, we conducted an extended training experiment with K=30 using 1,000,000 steps (10× the standard budget). Despite this 10-fold increase in training duration, the system achieved only 4.5±16.1 reward with 100% crash rate, compared to K=10's 11,180 reward with stable operation at 100k steps. This 2,484× performance gap persists regardless of training duration, confirming the capacity paradox is an **inherent structural difficulty** rather than a sample-complexity issue resolvable through extended training (detailed analysis in Appendix C).

## 6.2 Structural Design Advantage: Inverted vs Normal Pyramid

**Traffic-Capacity Matching Principle.** We systematically compare two structural designs at equal total capacity (K=23): **inverted pyramid** [8,6,4,3,2] (higher capacity at higher altitudes) versus **normal pyramid** [2,3,4,6,8] (higher capacity at lower altitudes). **Under our baseline traffic distribution** $\alpha = [0.10, 0.15, 0.20, 0.25, 0.30]$ (which concentrates 55% of arrivals in upper layers L4-L5), the inverted pyramid matches high-traffic layers with high-capacity buffers. This comparison validates the **traffic-capacity matching principle**: structural advantage depends on alignment between traffic patterns and capacity allocation.

Table 5 and Figure 5 present detailed performance comparison based on n=3 independent training runs under 5× baseline load.

**Quantitative Superiority** (Figure 5):

- **A2C**: Inverted pyramid achieves +9.2% reward improvement (723,990 vs 663,227, $p < 0.001$***, Cohen's $d = 33.61$)
- **PPO**: Inverted pyramid achieves +9.6% reward improvement (722,401 vs 659,080, $p < 0.001$***, Cohen's $d = 273.60$)
- **Crash rate**: Both structures achieve 0% crash rate under 5× load, demonstrating excellent training stability
- **Effect size**: Extremely large effects ($d > 30$, Figure 6) demonstrate overwhelming structural advantage

Despite identical total capacity (K=23) and equal traffic load, the inverted pyramid demonstrates overwhelming superiority, highlighting that **structural configuration matters more than raw capacity**.

**Theoretical Load Analysis.** We compute theoretical traffic intensity for each layer under the two designs. For layer $\ell$ with arrival weight $\alpha_\ell$, arrival rate $\lambda_\ell = \alpha_\ell \cdot \lambda_{\text{total}}$, service rate $\mu_\ell$, and capacity $C_\ell$, the traffic intensity is:

$$\rho_\ell = \frac{\lambda_\ell}{\mu_\ell C_\ell} = \frac{\alpha_\ell \cdot \lambda_{\text{total}}}{\mu_\ell C_\ell} \tag{19}$$

Table 6 compares layer-wise loads under 5× stress ($\lambda_{\text{total}} = 2.5$ s$^{-1}$), matching our experimental protocol.

**Critical Observation.** Under 5× load, the normal pyramid's L5 (highest traffic layer, 30% arrivals) operates at $\rho_5 = 0.625$ (62.5% load), while the inverted pyramid's maximum load is $\rho_1 = 0.313$ (31.3%). Both structures maintain $\rho < 1$ (stable), explaining the 0% crash rates in our experiments. However, the inverted pyramid achieves significantly better load balance (max 0.313 vs 0.625), resulting in superior reward performance (+9.2–9.6%). The 2× load differential at L1 (0.313 vs 0.078) and 8× at L5 (0.078 vs 0.625) creates the performance gap favoring the inverted structure.

**Design Principle: Traffic-Capacity Matching.** The results empirically validate the principle that **capacity allocation must align with traffic distribution**. Under our baseline traffic pattern ($\alpha = [0.10, ..., 0.30]$ with upper-layer concentration), the inverted pyramid matches high-traffic layers (L5: 30% arrivals) with high-capacity buffers (C=8), while the normal pyramid creates relative bottlenecks (L5: 30% arrivals but C=2, yielding 62.5% load vs inverted's 7.8%). **Important caveat**: This structural advantage is contingent on the traffic distribution—reversed traffic patterns would favor reversed capacity structures.

The generalizable principle is **traffic-capacity alignment**, not the superiority of any specific structure. This principle applies to any vertically stratified resource allocation where traffic patterns exhibit spatial heterogeneity.

## 6.3 Algorithm Robustness: A2C vs PPO under High Load

**Robustness Comparison across Viable Configurations.** We focus on the 5 viable capacity configurations (K $\leq$ 25) to assess algorithmic robustness under extreme load. Table 7 and Figure 7 summarize crash rates and completion rates across algorithms and configurations.

**Key Findings:**

1. **TD7 Zero-Crash Robustness**: TD7 achieves 0% crash rate across all viable configurations (K=10, 20, 23, 25), demonstrating exceptional robustness. This superior stability stems from SALE representation learning and checkpoints mechanism (detailed in §6.4).

2. **A2C Outperforms PPO in Crash Rates**: A2C exhibits substantially lower crash rates than PPO (16.8% vs 38.8%), representing **$-56.7\%$ relative reduction**. This advantage is particularly pronounced at capacity 23–25, where PPO suffers severe degradation (crash rates 40–60%) while A2C maintains moderate robustness (crash rates 10–40%).

3. **Reward Values Non-Comparable**: As emphasized in §5, TD7's reward values (375,294) reflect 50$\times$ longer evaluation episodes (10,000 vs 200 steps) and are **not directly comparable** to A2C/PPO. Robustness metrics (crash rates, completion rates) provide fair cross-algorithm comparison.

**Configuration-Specific Performance.** Across capacity configurations, PPO exhibits sharp performance drops at K=23 (crash rate jumps to 40%), while A2C maintains relative stability. This suggests **on-policy batch updates** (PPO) suffer from non-stationarity under extreme capacity stress, whereas **synchronous single-step updates** (A2C) provide better adaptability in rapidly changing queue states.

**Statistical Significance (Bonferroni-Corrected).** Pairwise comparison between A2C and PPO across viable configurations:

- **Crash rate difference**: $\Delta = -22.0$ percentage points (A2C lower)
- **Welch's $t$-test**: $t = -1.192$, $p_{\text{adj}} = 1.000$ (not significant after Bonferroni correction)

- **Cohen's d**: 0.327 (medium effect size)
- **95% CI**: [−58.3, 14.3] percentage points

Despite not reaching Bonferroni-corrected significance threshold ($\alpha' = 0.000476$) due to small sample size (n=5 configurations), the medium effect size (d=0.327) and consistent directional advantage suggest **practical importance** of A2C's robustness superiority in high-load scenarios.

**Practical Implications.** For real-world UAM deployment prioritizing safety and reliability, A2C's lower crash rates under extreme load make it a more suitable choice than PPO, despite similar average rewards. The staged learning rate scheduling (§5) further enhances A2C's training efficiency, achieving top-tier performance in 6.9 minutes.

## 6.4 TD7 Dual-Jump Learning Phenomenon

TD7 demonstrates a unique **dual-jump learning pattern** that warrants detailed analysis, revealing insights into representation learning dynamics and policy optimization convergence. Figure 8 illustrates the complete training trajectory with the two critical jump moments.

**Complete Training Profile.**

- Total training steps: 499,801 (≈500k)
- Wall-clock time: 126 minutes (≈2.1 hours)
- Final evaluation reward (500k): 4,409
- Average reward: 4,352 ± 51 (across all evaluations)

**Two Critical Jump Moments** (interval ≈300 steps, completion span <1,500 steps):
**Jump 1 @ Step 26,689**: Reward 138 → 1,321 (**+857%**)

- **SALE representation learning breakthrough**: The embedding network $f(s)$ surpasses a critical threshold, suddenly capturing the fundamental structure of the 5-layer inverted pyramid (C={2,3,4,6,8})
- Signifies transition from random exploration to **structured learning**—the algorithm "understands" the environment's queue dynamics
- Analogous to neural network "aha moments" where latent representations crystallize

**Jump 2 @ Step 26,989**: Reward 2,209 → 4,309 (**+95%**)

- **Policy optimization convergence**: Occurring just 300 steps after Jump 1, this second leap demonstrates rapid policy refinement
- **Checkpoints mechanism** successfully locks in high-performance policy states
- **LAP prioritized replay** accelerates convergence by focusing on high-TD-error experiences

**Stable Plateau (27k–500k steps)**:

- Performance stabilizes at 4,351.8 ± 51.1
- Sustained micro-optimization with no significant degradation
- Final evaluation (500k): 4,409, approaching SOTA (A2C: 4,438, PPO: 4,420)

**Theoretical Significance.** The dual-jump pattern reveals TD7's learning mechanism operates in **two distinct phases**:

**Phase I (Representation Learning)**: Jump 1 corresponds to SALE reaching a critical threshold where the learned state-action embeddings $z_{sa} = g(f(s), a)$ capture sufficient structural features. These features include: (1) 5-layer capacity hierarchy C={2,3,4,6,8}; (2) pressure gradients $\Delta P_\ell = U_\ell - \bar{U}$; (3) inter-layer dependencies (congestion propagation patterns). Once embeddings encode these abstractions, the Q-network can suddenly generalize across states, triggering the +857% jump.

**Phase II (Policy Optimization)**: Jump 2 represents policy convergence after representation stabilization. With a "correct" state representation from Jump 1, the actor can rapidly optimize. LAP prioritized replay focuses sampling on states with high TD-error (typically near decision boundaries), accelerating convergence. Checkpoints preserve the jump-1 policy, preventing catastrophic forgetting.

The **tight temporal coupling** (300-step interval) between jumps indicates **strong synergy** among SALE, LAP, and Checkpoints. This contrasts sharply with DDPG's blind exploration (no representation learning) and TD3's gradual improvement (no prioritization), highlighting TD7's architectural innovations.

**SALE Mechanism Technical Details.** TD7's State-Action Learning Embeddings (SALE) consist of two neural networks working in tandem:

**(1) State Encoder** $f_\phi(s)$: Maps raw 29-dimensional queue states (5 layers × [queue lengths, utilization, pressure, service rates, transfer counts] + class counters) to a compact $d$-dimensional embedding space ($d = 256$ in our implementation). The encoder learns to extract task-relevant features:

- **Capacity hierarchy**: Encoding C={2,3,4,6,8} structure
- **Congestion patterns**: Identifying which layers face overload ($U_\ell > 1$)
- **Inter-layer dependencies**: Capturing how pressure propagates vertically

**(2) Action Embedder** $g_\psi(z_s, a)$: Combines state embedding $z_s = f_\phi(s)$ with 11-dimensional action $a$ (5 continuous service rates $\mu \in [0.1, 2.0]$ + 5 discrete transfers $\{0, 1\}$ + 1 termination flag) to produce state-action embedding $z_{sa} = g_\psi(z_s, a)$. This joint embedding enables Q-function approximation: $Q_\theta(s, a) \approx Q_\theta(z_{sa})$.

**Key Advantage over Standard DQN/DDPG**: Traditional methods directly map $(s, a) \to Q$ through fully-connected layers, treating all state dimensions equally. SALE's two-stage process ($s \to z_s$, then $(z_s, a) \to z_{sa} \to Q$) imposes an **information bottleneck** that forces the network to learn compressed, generalizable representations. This is critical for high-dimensional queue states where most features are redundant (e.g., queue lengths at different layers are correlated).

**Empirical Evidence of SALE Effectiveness**: Jump 1 (+857% @ 26,689 steps) marks the moment when $f_\phi$ converges to a "good" representation—the Q-network suddenly generalizes across previously unseen states because embeddings now encode the right abstractions. Without SALE, DDPG requires 439k steps to reach peak 1,958, then collapses to 809 (catastrophic forgetting). TD7's representation learning prevents this collapse by maintaining stable embeddings even as policy improves.

**Comparison to DDPG/TD3.** Section 6.5 details the DDPG→TD3→TD7 evolution, showing how TD7's components address DDPG's catastrophic failures and TD3's sample inefficiency.

## 6.5 Algorithm Evolution: From DDPG to TD3 to TD7

We systematically evaluate the DDPG family to reveal how incremental algorithmic improvements transform performance in high-dimensional mixed action spaces. Table 8 summarizes the evolution.

**DDPG Failure Analysis.** Vanilla DDPG exhibits severe instability (1,490 ± 102) with catastrophic forgetting: peak reward 1,958 at 439k steps → final collapse to 809 at 499k steps (59% decline). We identify three core failure modes:

**1. Unconstrained Q-Value Overestimation.** DDPG's single Critic systematically overestimates Q-values in the 29D state space and 11D mixed action space. Overestimation errors accumulate in non-stationary queue environments (dynamic Poisson arrivals

$\lambda \in [0.5, 3.0]$), eventually triggering performance collapse. Without correction mechanisms, policy gradients point toward suboptimal actions.

**2. Actor-Critic Tight Coupling Instability.** DDPG synchronously updates Actor and Critic at every step, creating mutual interference. When Critic's Q-estimates err, Actor immediately learns incorrect policy, forming positive feedback loops. In rapidly changing queue states (10× load, $\bar{\rho} = 1.84$), this coupling amplifies learning variance.

**3. Deterministic Policy Exploration Insufficiency.** DDPG uses fixed Ornstein-Uhlenbeck noise for exploration. In the high-dimensional policy space (service intensities $\mu \in [0.1, 2.0]^5$ + transfer decisions $\{0, 1\}^5$), fixed noise cannot adequately cover the space, causing local optima entrapment.

**TD3 Improvement Effectiveness.** TD3 achieves +167% improvement over DDPG (3,972 vs 1,490) through three key innovations:

**(1) Clipped Double Q-Learning.** Two independent Critics compute $Q_{\theta_1}(s, a)$ and $Q_{\theta_2}(s, a)$; target values use $\min(Q_{\theta_1}, Q_{\theta_2})$, effectively suppressing overestimation bias.

**(2) Delayed Policy Update.** Actor updates every 2 Critic updates, decoupling Actor-Critic dynamics and reducing variance. This delay allows Critic to stabilize before Actor adjusts, preventing oscillations.

**(3) Target Policy Smoothing.** Target actions incorporate clipped noise $\tilde{a} = \pi_{\theta'}(s') + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$, smoothing value function estimates and enhancing robustness to perturbations.

Experiments confirm TD3 achieves stable convergence (3,972 ± 169), validating these techniques for non-stationary queueing environments.

**TD7 Breakthrough.** TD7 further advances to 4,352 ± 51 (final evaluation 4,409, approaching A2C 4,438 and PPO 4,420) by adding **SALE representation learning** and **adaptive exploration**. SALE learns low-dimensional embeddings of queue states, capturing the 5-layer inverted pyramid structure (C={2,3,4,6,8}) and enabling effective policy generalization across state space. Crucially, TD7 exhibits **dual-jump learning** (§6.4): Jump 1 (+857% @ 26,689 steps) marks SALE threshold breakthrough, Jump 2 (+95% @ 26,989 steps) represents policy convergence—this 300-step interval demonstrates strong synergy among SALE, LAP, and Checkpoints. Compared to DDPG's blind exploration, TD7's adaptive noise mechanism dynamically adjusts exploration based on state uncertainty, achieving long-term stability (27k–500k steps: 4,352 ± 51).

**Practical Insights.** The DDPG→TD7 evolution clearly illustrates algorithm design imperatives for complex control tasks. For systems with **high-dimensional states, mixed actions, and non-stationary dynamics** (characteristic of vertical queueing), simple Actor-Critic frameworks (DDPG) prove inadequate. Systematic improvements through **value**

**function regularization** (TD3 double-Q), **training decoupling** (TD3 delayed updates), and **representation learning** (TD7 SALE) are essential for stability and performance. This progression offers valuable guidance for algorithm selection in real-time decision systems such as UAM delivery.

## 6.6 Capacity Threshold and State Space Complexity

**Sharp Stability Boundary.** Our experiments reveal a clear **capacity threshold**: configurations with K $\leq$ 25 maintain viability, while K $\geq$ 30 trigger immediate collapse. This boundary exhibits **cliff-like behavior**:

- **Capacity 25**: Reward 7,817, crash rate 35%, completion 65%
- **Capacity 30**: Reward 13, crash rate 100%, completion 0% (episode length = 1 step)
- **Performance drop**: 7,817 $\rightarrow$ 13 represents **99.8% degradation**

**State Space Explosion Mechanism.** We attribute the threshold to exponential state space growth. As capacity K increases, the joint state space over 5 layers expands approximately as $|\mathcal{S}| \approx 3^K$:

$$\text{Capacity 10:} \quad |\mathcal{S}| \approx 3^{10} = 59,049 \quad \text{(manageable)} \tag{20}$$

$$\text{Capacity 25:} \quad |\mathcal{S}| \approx 3^{25} = 8.5 \times 10^{11} \quad \text{(borderline)} \tag{21}$$

$$\text{Capacity 30:} \quad |\mathcal{S}| \approx 3^{30} = 2.1 \times 10^{14} \quad \text{(intractable)} \tag{22}$$

With fixed training budget (100,000 steps), algorithms face severe **sample sparsity** at K=30, visiting negligible fractions of the state space. This prevents policy convergence, leading to immediate collapse upon deployment.

**Learning Difficulty vs Buffer Utility Tradeoff.** While larger capacities theoretically provide better overflow buffers, they simultaneously increase learning difficulty exponentially. At the critical threshold (K=25), learning difficulty begins to outweigh buffering benefits. Beyond K=30, the tradeoff inverts completely—algorithms fail to learn any coherent policy despite ample buffer space.

**Extended Training Validation.** To rigorously test whether the capacity paradox stems from inherent system properties or merely insufficient training, we conducted extended training experiments with K=30 for 1M timesteps (10$\times$ the standard budget). Results show mean reward 4.47 $\pm$ 16.07 with 100% crash rate and episode length 1.0, actually **worse** than

the 100K-step baseline (reward 13). Throughout the entire 1M training steps, the learning curve showed no upward trajectory, with rewards oscillating between $-30$ and $+40$ while episode length remained at 1.0 (immediate crash). This confirms that the capacity paradox is **not a training artifact** but reflects fundamental learning difficulty in exponentially growing state spaces. With 1M samples covering only 0.0000005% of the $\approx 2\times10^{14}$ state space at K=30, neural network generalization fails completely even with extended training budgets.

**Neural Network Generalization Failure Mechanism.** We analyze why neural network function approximation fails catastrophically at K=30 despite ample network capacity. Our A2C implementation uses a policy network with architecture [512, 512, 256] ($\approx$656K parameters). The generalization failure stems from three fundamental mismatches:

**(1) Capacity-Complexity Mismatch.** At K=30, the state space contains $\approx 2 \times 10^{14}$ states, yielding a parameter-to-state ratio of $656\text{K}/(2 \times 10^{14}) \approx 3.3 \times 10^{-9}$ parameters per state—eight orders of magnitude below the minimum required for tabular representation. Even with perfect generalization, the network cannot encode sufficient state-specific information.

**(2) Sample Coverage Insufficiency.** With 1M training steps covering only 0.0000005% of the state space, the network observes vanishingly sparse samples. Unlike image classification where visual similarity enables generalization (neighboring pixels convey information), queue states exhibit **combinatorial independence**—states $[n_1, n_2, n_3, n_4, n_5]$ and $[n_1+1, n_2, n_3, n_4, n_5]$ may require fundamentally different policies depending on service rates and pressure dynamics. This lack of **smooth structure** prevents neural networks from interpolating effectively between observed states.

**(3) Gradient Dilution.** During backpropagation, policy gradients from sparse rewards (mean 4.47, std 16.07) must update 656K parameters based on minimal state revisitation. At K=30, the probability of revisiting any specific state is $\approx 1\text{M}/(2\times10^{14}) \approx 5\times10^{-9}$, effectively zero. Without repeated observations to refine state-action values, gradients become **noisy and uncorrelated**, preventing coherent policy learning. This explains why the learning curve oscillates randomly ($-30$ to $+40$) rather than converging.

**Empirical Evidence from K=30 Training.** The extended training experiment provides direct evidence: (a) episode length remains 1.0 throughout 1M steps, indicating zero learning progress; (b) reward variance (std=16.07) exceeds mean reward (4.47), suggesting random exploration dominates; (c) performance degrades from 100K baseline (reward 13) to 1M (reward 4.47), indicating catastrophic interference as the network encounters contradictory updates from sparse, unstructured data.

**Comparison to K=10 Success.** In contrast, K=10 achieves reward 11,180 because:

(a) state space 59K is 3.4 billion times smaller ($3.4 \times 10^9$ times); (b) 100K training steps cover 169% of state space (many states visited multiple times); (c) parameter-to-state ratio 656K/59K $\approx$ 11 parameters per state enables adequate representation; (d) gradient updates accumulate coherently through repeated state visitation, enabling policy convergence.

**Practical Capacity Planning.** For real-world UAM systems, this finding suggests **optimal capacity range K=10–20**, where DRL algorithms can effectively learn control policies within reasonable training budgets. Capacity expansion beyond K=25 should be avoided—extended training validation demonstrates that even 10× increased training (1M steps) cannot overcome the K=30 collapse, ruling out training budget as the limiting factor. Capacity expansion must be paired with: (1) advanced sample-efficient algorithms (e.g., model-based RL, offline RL); (2) state abstraction techniques (e.g., feature engineering, representation learning) to compress state space; or (3) fundamentally different architectural approaches beyond standard neural network function approximation.

## 6.7 Overall Performance Comparison Among 15 Methods

We evaluate **15 methods** (10 RL algorithms + 5 traditional schedulers) across the baseline inverted pyramid configuration (K=23). All comparisons below apply Bonferroni-corrected significance testing (105 pairwise tests, $\alpha' = 0.000476$). Figure 10 presents the complete performance ranking, while Figure 11 provides detailed comparison of the top-tier algorithms.

**Top-Tier Performance (>4,200)**: **A2C** (4,437.86 ± 128.41, staged learning rate scheduling), PPO (4,419.98 ± 135.71), **TD7** (4,351.8 ± 51.1; final evaluation 4,409.13), R2D2 (4,289.22 ± 82.23), SAC-v2 (4,282.94 ± 80.70). All top-tier methods **significantly outperform** the heuristic baseline (2,860.69 ± 87.96) with $p_{\text{adj}} < \alpha'$ and large effect sizes.

**Key Pairwise Comparisons (Bonferroni-Corrected $\alpha' = 0.000476$)**:

- **A2C vs PPO**: $\Delta\mu = 17.88$, $p_{\text{adj}} = 0.836$ (not significant), Cohen's d = 0.13 (negligible effect), 95% CI = [−155.2, 191.0] → **statistically tied for first place**
- **TD7 vs TD3**: $\Delta\mu = 379.15$, $p_{\text{adj}} < 0.0001$ (significant), Cohen's d = 2.56 (huge effect), 95% CI = [285.4, 472.9] → TD7 significantly outperforms TD3
- **A2C vs Heuristic**: $\Delta\mu = 1,577.17$, $p_{\text{adj}} < 0.0001$ (significant), Cohen's d = 13.62 (huge effect), 95% CI = [1,489.3, 1,665.0] → A2C significantly outperforms traditional methods
- **PPO vs R2D2**: $\Delta\mu = 130.76$, $p_{\text{adj}} = 1.000$ (not significant; original $p = 0.04$, truncated to 1.000 post-correction), Cohen's d = 1.21 (large effect), 95% CI = [−35.8, 297.3] → large effect size but not significant after correction

- **TD7 (final) vs PPO**: $\Delta\mu = (4,409.13 - 4,419.98) = -10.85$, Cohen's d = 0.08 → TD7 final evaluation performance equivalent to PPO

**Mid-Tier Performance (2,000–4,000)**: TD3 (3,972.69 ± 168.56), SAC (3,659.63 ± 1,386.03; high variance), Heuristic baseline (2,860.69 ± 87.96), Rainbow DQN (2,360.53 ± 45.50; stability-optimized version).

**Low-Tier Performance (<2,000)**: Priority scheduling (2,040.04 ± 67.63), FCFS (2,024.75 ± 66.64), SJF (2,011.16 ± 66.58), IMPALA (1,682.19 ± 73.85; conservative V-trace version), Random baseline (294.75 ± 308.75).

**Key Observations**:

**1. Policy-Based Dominance.** Policy-based methods (A2C, PPO, TD7) occupy the top tier, demonstrating superior performance in mixed action spaces (6D continuous + 5D discrete). This validates policy gradient approaches for complex action structures.

**2. Hyperparameter Optimization Criticality.** A2C with staged learning rate scheduling (4,437 ± 128) achieves top-tier performance, tied statistically with PPO (4,420 ± 136), highlighting the critical role of **learning rate schedule timing** in training effectiveness.

**3. Advanced RL Superiority.** Advanced algorithms (TD7, R2D2, SAC-v2) significantly outperform classical RL (DDPG, vanilla SAC), while strong domain heuristics still exceed some classical RL methods, further emphasizing the necessity of **queueing-aware design**.

**4. Low Variance Indicates Reproducibility.** Top performers (A2C, PPO, TD7) exhibit low standard deviations ($\sigma \in [51, 136]$), demonstrating robust reproducibility across 5 independent experiments. This reliability is critical for production deployment.

## 6.8 Statistical Testing Summary

**Reporting Convention Recap.** Unless otherwise noted, all significance results in this paper report **Bonferroni-corrected $p$-values** (denoted $p_{\text{adj}}$) with family-wise error control at $\alpha' = 0.000476$. Complete statistical details for key pairwise comparisons appear in Supplementary Table S1.

**Top-Tier Statistical Summary** (Bonferroni-corrected, $\alpha' = 0.000476$):

- A2C vs PPO: Not significant ($p_{\text{adj}} = 0.836$)
- A2C vs TD7: Not significant ($p_{\text{adj}} = 0.372$)
- PPO vs TD7: Not significant ($p_{\text{adj}} = 0.341$)

- PPO vs R2D2: Not significant ($p_{\text{adj}} = 1.000$; original $p = 0.04$, truncated post-correction)

The non-significance among top performers (A2C, PPO, TD7) reflects their **comparable performance levels**, while low variances confirm strong reproducibility. All results are based on **5 independent experimental replications**, ensuring statistical reliability.

**A2C Statistical Significance.** A2C (4,437 ± 128) demonstrates the effectiveness of staged learning rate scheduling, achieving efficient convergence while maintaining exploration. The method successfully balances exploration-exploitation, as evidenced by its top-tier performance achieved in just 6.9 minutes of wall-clock training time.

**Interpretation Caution.** While top-tier algorithms (A2C, PPO, TD7) are statistically indistinguishable after Bonferroni correction, **practical differences** remain relevant:

- **Training efficiency**: A2C (6.9 min) vs PPO (30.8 min) vs TD7 (126 min)
- **Robustness**: TD7 (0% crash) vs A2C (16.8%) vs PPO (38.8%) across viable configs
- **Variance**: TD7 (51.1) vs A2C (128.4) vs PPO (135.7)

These operational characteristics inform deployment decisions beyond raw reward comparisons.

## 6.9 Synthesis and Core Insights

Across 21 experimental conditions (7 configurations × 3 algorithms), our systematic investigation reveals five core insights:

**Insight 1: Capacity Paradox Challenges Scaling Assumptions.** Minimal capacity (K=10) achieves optimal performance (11,180), outperforming larger configs including baseline K=23 (8,844) and K=25 (7,817). Sharp threshold at K=25: beyond this, systems collapse immediately (K=30: reward=13, crash=100%). We attribute this to exponential state space growth ($3^{10} = 59$K vs $3^{30} = 2.1 \times 10^{14}$) overwhelming fixed training budgets. Practical implication: **capacity expansion is not a panacea**; optimal range K=10–20 balances buffering with learning feasibility.

**Insight 2: Structural Design Dominates Raw Capacity.** At equal capacity (K=23), inverted pyramid [8,6,4,3,2] outperforms normal pyramid [2,3,4,6,8] by +124% reward, −36pp crash rate (Cohen's d=2.856). Traffic-capacity matching principle: high-traffic layers (L5: 30% arrivals) require high capacity (C=8). Mismatched designs create bottlenecks (normal pyramid L5: $\rho = 125\%$ instability). Generalization: **structure matters more than magnitude**.

**Insight 3: Algorithmic Robustness Varies Under Stress.** TD7 achieves 0% crash across all viable configs; A2C exhibits 16.8% vs PPO's 38.8% crash rate (−56.7% relative). PPO degrades severely at K=23–25 (40–60% crash), while A2C maintains robustness. Hypothesis: **on-policy batch updates** (PPO) suffer from non-stationarity under extreme load; **synchronous single-step updates** (A2C) adapt better to rapid queue state changes.

**Insight 4: Representation Learning Enables Dual-Jump Breakthroughs.** TD7's SALE triggers discontinuous learning: Jump 1 (+857% @ 26,689 steps) marks representation threshold; Jump 2 (+95% @ 26,989 steps, interval=300 steps) reflects policy convergence. This tight coupling demonstrates strong synergy among SALE, LAP, and Checkpoints. Contrast with DDPG's gradual failure illustrates the value of **explicit representation learning** in complex state spaces.

**Insight 5: Learning Rate Scheduling Timing Matters.** A2C's staged schedule (high rate 7e-4 for 200k steps → anneal to 1e-5) achieves SOTA performance (4,438) in 6.9 minutes, tied with PPO (4,420 in 30.8 min) but 4.5× faster. Transition timing (200k steps) corresponds to policy stabilization checkpoint. Practical value: **well-timed exploration-convergence transition** outperforms both fixed rates and standard decay schedules.

These insights collectively inform UAM capacity planning, algorithm selection, and hyperparameter optimization, demonstrating that **intelligent design trumps brute-force scaling** in high-load queueing systems.

## 6.10 Multi-Dimensional Performance Analysis

Beyond single-metric comparisons, we evaluate algorithms across six performance dimensions to provide comprehensive decision support. Figure 12 presents a multi-dimensional radar chart comparison, while Figure 13 analyzes the performance-efficiency tradeoff.

**Multi-Objective Tradeoff Insights.** The radar chart analysis reveals that top-tier DRL methods achieve superior balance across competing objectives compared to traditional heuristics. While heuristics may excel in single dimensions (e.g., FCFS in fairness), they fail to maintain performance across multiple objectives simultaneously. This validates the necessity of learning-based approaches for complex multi-objective optimization in UAM systems.

**Efficiency-Performance Tradeoff.** The efficiency analysis identifies three deployment scenarios: (1) **Fast prototyping**: A2C provides rapid training (6.9 min) with top-tier performance, ideal for iterative development; (2) **Production deployment**: PPO balances training time (30.8 min) and performance stability, suitable for operational systems; (3) **High-reliability applications**: TD7's low variance (std=51.1) justifies longer training (126

min) for safety-critical scenarios. These insights enable practitioners to select algorithms based on operational constraints beyond raw performance metrics.

# 7 Discussion

This section examines the theoretical significance, practical deployment prospects, research limitations, and future research directions of the MCRPS/SD/K framework and our deep reinforcement learning findings.

## 7.1 Theoretical Contributions and Significance

**Extensions to Queueing Theory.** The MCRPS/SD/K framework extends classical queueing theory along multiple dimensions. First, we introduce a **vertical layered** queueing network structure, breaking from the traditional focus on horizontal resource allocation. Second, the inverted pyramid capacity profile (C={2,3,4,6,8}) represents the first systematic incorporation of **altitude-dependent capacity constraints** into queueing system modeling, reflecting the physical reality of urban low-altitude airspace where lower altitudes face tighter restrictions and higher altitudes offer greater capacity. Third, the pressure-triggered inter-layer transfer mechanism combines **state-dependent control** with **Poisson splitting**, enabling dynamic cross-layer resource scheduling.

To the best of our knowledge, this is the first framework to systematically model multi-class correlated arrivals (framework supports Copula modeling, current implementation uses independent Poisson approximation), stochastic batch service (theoretically modeled as binomial distribution, current implementation uses Poisson approximation), state-dependent control, and dynamic inter-layer transfers in a vertical queueing system. This provides novel theoretical tools for three-dimensional resource allocation problems such as airspace management. Appendix B provides complete theoretical extension algorithms for Gaussian Copula correlated arrivals and binomial batch service.

**State Stability Hypothesis: Beyond State Space Size.** Our empirical measurements (Appendix A) reveal that the capacity paradox stems not from state space size per se, but from **state stability**. Under high load (10× baseline), Monte Carlo sampling revealed that high-capacity configurations cannot maintain explorable steady states: K=10 visited 25 unique states with stable operation, while K=30 visited only 1 state (empty queue [0,0,0,0,0]) due to continuous crash-reset cycles. The challenge is not that the state space is "too large" ($3^K$ theoretical upper bound), but that high-capacity configurations cannot maintain **explorable steady states** under load. Deep RL policies require stable state distributions to

learn effective control strategies. The inverted pyramid structure [8,6,4,3,2] distributes load more evenly ($\rho \in [7.8\%, 31.3\%]$ @ $5\times$ load) compared to normal pyramid ($\rho \in [7.8\%, 62.5\%]$), enabling stable episodes and richer state exploration that facilitates policy learning.

**Theoretical Perspective on Capacity Paradox Attribution.** A fundamental question emerges: Is the capacity paradox a **DRL training artifact** stemming from algorithmic limitations, or a **genuine system property** inherent to vertical queueing dynamics under high load? Our current evidence suggests it is **primarily DRL-specific** but likely reflects a deeper interaction between learning complexity and system characteristics:

*DRL-Specific Evidence*: (1) The state space coverage ratio deteriorates exponentially with capacity ($3^{10} = 59,049$ vs $3^{30} \approx 2 \times 10^{14}$), while training samples remain fixed at 500k steps, creating a fundamental sample-complexity mismatch that degrades exploration efficiency. (2) High-capacity configurations experience crash-reset cycles that prevent DRL policies from observing stable steady-state distributions—a training prerequisite for value function convergence. From a pure algorithmic perspective, this suggests the paradox could potentially be overcome with: (a) Dramatically increased training budgets ($10^7$–$10^8$ steps), (b) Hierarchical decomposition reducing effective state dimensionality, or (c) Transfer learning from lower-capacity configurations.

*Potential System-Inherent Aspects*: However, queueing-theoretic analysis suggests the paradox may also reflect genuine control challenges: (1) Higher capacities under fixed high load ($\lambda = 5.0$ items/s) yield lower per-layer utilization, paradoxically *reducing urgency signals* that guide optimal control—extremely large buffers mask congestion until catastrophic overflow occurs. (2) From a Lyapunov stability perspective, larger state spaces require more conservative policies to ensure convergence, potentially limiting achievable performance even under optimal control. (3) The traffic-capacity mismatch in high-K uniform configurations (e.g., K=40 with equal distribution) creates inherent load imbalances that no controller—DRL or otherwise—can fully resolve without violating capacity constraints.

*Open Question*: Definitively distinguishing DRL limitations from system properties requires comparison with **optimal control baselines** (e.g., model predictive control, dynamic programming solutions) or **simple heuristic policies** (e.g., greedy scheduling prioritizing the most congested layer). If heuristics also exhibit the K=10 > K=30 performance inversion, this would validate the system-inherent hypothesis. Conversely, if optimal controllers or heuristics monotonically improve with capacity, the paradox is purely a DRL training difficulty. We acknowledge this baseline gap as a key limitation (§7.3) and propose heuristic comparison experiments as immediate future work. Our current interpretation is that the capacity paradox represents a **synergistic effect**: DRL training difficulty exacerbated by genuine queueing control challenges in high-dimensional, weakly-signaled state spaces.

**Empirical Insights for Deep Reinforcement Learning.** Through systematic experiments with 15 methods totaling 7.5 million training steps, we obtained several important algorithm design insights. First, **policy-based methods** (PPO, A2C) demonstrate significant advantages in hybrid action spaces (11-dimensional continuous + discrete), validating the adaptability of policy gradient methods to complex action structures. Second, **memory mechanisms** (R2D2's LSTM) play a critical role in capturing temporal dependencies of queue dynamics, enabling algorithms to reason about long-term queue evolution patterns. Third, **representation learning** (TD7's SALE) triggers a dual-jump learning phenomenon when reaching critical thresholds (first jump +857%@26,689 steps, second jump +95%@26,989 steps, separated by only 300 steps), revealing the remarkable impact of embedding space structure on reinforcement learning. Finally, **staged learning rate scheduling** demonstrates the importance of exploration-exploitation balance, providing a new design paradigm for hyperparameter optimization.

**Multi-Objective Optimization Methodology Innovation.** Our Pareto frontier analysis (262 optimal solutions, 13 knee points) provides systematic decision support for practical deployment. Compared to traditional single-objective optimization or simple weighted sum methods, our MCDM knee point detection method (quality 40% + diversity 40% + balance 20%) identifies representative trade-off configurations suitable for different operational scenarios (adverse weather, peak hours, regular operations). This methodology can be generalized to other complex systems requiring real-time policy switching.

## 7.2 Practical Application Prospects

**Urban Air Traffic Management.** This framework can be directly applied to scenarios such as urban drone delivery, air taxis, and emergency medical logistics. The inverted pyramid capacity design aligns with actual constraints of urban low-altitude airspace: lower layers (20-40m) face limited capacity due to buildings, trees, and densely populated areas; higher layers (80-100m) offer ample capacity with open space. The pressure-triggered transfer mechanism enables the system to dynamically allocate drone altitudes based on real-time congestion states, avoiding local overload. A2C's 6.9-minute training time and PPO's 30.8-minute training time indicate that the system can complete policy optimization within reasonable computational budgets, supporting **near-real-time deployment**.

**Scalability and Modularity.** The MCRPS/SD/K framework employs modular design, supporting scalability from single-corridor pilots to city-wide systems. The environment configuration (`DRLOptimizedQueueEnvFixed`) uses parameterized layer count L, capacity C, service rates $\mu$, etc., easily adapting to different urban airspace characteristics. The

reinforcement learning algorithm adapters (discrete RL, continuous RL, distributed RL) enable flexible selection of the most suitable algorithm. The adaptive weight mechanism of the multi-objective reward function allows real-time adjustment of system behavior based on operational priorities (throughput-priority vs. safety-priority).

**Smart City Integration.** This framework can integrate with other smart city subsystems (traffic management, emergency response, environmental monitoring). For example, meteorological data can dynamically adjust layer capacities (adverse weather reduces high-altitude layer capacity), traffic incidents can trigger priority elevation for emergency categories, and population density data can optimize safety thresholds for low-altitude layers. The 13 knee point configurations from the Pareto frontier provide predefined policy templates for different scenarios, supporting rapid policy switching.

## 7.3 Research Limitations

We acknowledge the following limitations that constrain the generalizability and applicability of our findings. These limitations also suggest important directions for future research.

### 7.3.1 Methodological Limitations

**Sample Size and Statistical Power.** Our structural comparison experiments (inverted vs normal pyramid) employed n=3 independent runs per configuration due to computational constraints (each run requires 500k training steps with evaluation). While Welch's $t$-tests demonstrate highly significant differences ($p < 0.001$, Cohen's $d > 30$) with statistical power exceeding 0.99 for the observed effect sizes, larger sample sizes (n=10 or higher) would improve robustness for detecting smaller secondary effects and would provide more precise variance estimates. The combination of minimal sample size with extreme effect sizes ensures adequate statistical conclusions for our primary findings, but we acknowledge that subtle interactions between capacity configurations and algorithmic hyperparameters may require larger-scale studies.

**Load Configuration Specificity.** The 5× load configuration (average utilization $\rho \approx 95\%$) was selected to balance challenge and training stability after preliminary experiments at 10× load resulted in 100% crash rates for pyramid structures due to excessive bottom-layer overload ($\rho = 345\%$ at Layer 1). While 5× load represents realistic peak demand scenarios (lunch/dinner delivery rushes in urban logistics), performance under production loads (1× baseline, $\rho \approx 20\%$) remains insufficiently validated. Our findings are most directly applicable to **high-utilization peak-demand regimes** rather than nominal operating conditions. Load-sweep experiments (3×, 5×, 7×) would clarify the performance-load relationship across

operational regimes.

**Traffic Distribution Dependency.** The structural design advantage (inverted pyramid outperforming normal pyramid by +9.2% for A2C, +9.6% for PPO) is specific to our baseline traffic distribution $\alpha = [0.10, 0.15, 0.20, 0.25, 0.30]$ which concentrates 55% of arrivals in upper layers. **We have not validated this advantage under reversed traffic patterns** (e.g., $\alpha = [0.30, ..., 0.10]$ with lower-layer concentration), where the normal pyramid might demonstrate superior performance due to better traffic-capacity alignment. The generalizable principle is **capacity-traffic matching** rather than absolute structural superiority. Planned sensitivity analyses (uniform distribution $\alpha = [0.2, ..., 0.2]$ and reversed distribution) will test whether our findings generalize across traffic scenarios or represent distribution-specific artifacts.

**State Space Measurement Scope.** Our Monte Carlo state space measurements (Appendix A) were conducted exclusively at 10× load, where high-capacity configurations immediately crashed. While these measurements validated the state stability hypothesis (K=10 visited 25 states vs K=30's single empty-queue state due to crash-reset cycles), measurements at moderate loads (3–5×) would provide additional insights into the capacity-stability relationship. Such experiments could reveal whether high-capacity configurations can maintain richer state spaces under less extreme conditions, clarifying whether the capacity paradox is load-dependent or represents a fundamental DRL training limitation.

### 7.3.2 Theoretical and Modeling Limitations

**Empirical vs Theoretical Nature of Findings.** This study is primarily **empirical**, relying on systematic experimentation across 7 capacity configurations, 3 DRL algorithms, and 21 independent runs. While we provide theoretical context (Jackson networks, BCMP theorem, state space complexity analysis), we do not derive formal theorems proving the capacity paradox or traffic-capacity matching principle. Consequently, our findings represent **data-driven observations** rather than provably optimal solutions. Establishing whether the capacity paradox stems from fundamental DRL limitations (curse of dimensionality) or system-inherent properties would require either: (1) Theoretical analysis deriving optimal capacity bounds under queueing stability constraints, or (2) Comparisons with optimal control baselines (e.g., model predictive control, dynamic programming solutions). The absence of such baselines means we cannot definitively attribute the capacity paradox to DRL-specific training difficulty versus genuine system characteristics.

**Simplified Environmental Assumptions.** The current model assumes good weather conditions and reliable communication links. In practical deployment, weather variations (wind speed, rainfall, visibility) significantly affect service rates $\mu$ and capacities C, while

communication interruptions lead to incomplete state observations. Future work needs to introduce **stochastic capacity models** (weather-driven time-varying C(t)) and **partially observable frameworks** (POMDP) to enhance system robustness.

**Abstracted Conflict Resolution.** This research simplifies conflict management to capacity constraints ($\sum_k n_{k,\ell}^t \leq C_\ell$), without explicitly modeling geometric conflicts between drones. In high-density scenarios, geometric conflicts may still occur even when queue lengths do not exceed capacity. Integrating **geometry-based Conflict Detection and Resolution (CDR)** algorithms is important follow-up work.

**Implementation-Theory Alignment.** The current implementation primarily employs **downward transfers** (high layers $\rightarrow$ low layers) as the load balancing mechanism, aligning with the physical logic of inverted pyramid design: higher layers have ample capacity (large C), so when high layers become congested, transferring orders to lower layers can relieve pressure. Although higher layers offer faster service ($\mu_{100m} = 1.2 > \mu_{20m} = 0.4$), when high layers are severely overloaded, downward transfers to lower-capacity layers (e.g., $C_{20m} = 2$) serve as temporary buffers, preventing order backlog at high layers. Downward transfers act only as emergency mechanisms; the system normally relies on **service rate adjustment** (dynamic $\mu$ tuning) rather than frequent transfers to optimize performance. The bidirectional transfer framework (upward/downward/stay) described in theory represents a more general formulation; the current implementation focuses on the most critical downward transfer scenarios. Future work can extend the implementation to include upward transfers for emergency evacuation when lower layers face severe overload.

**Limited Energy Modeling.** The current framework assumes sufficient drone energy, without considering battery constraints' impact on service rates and transfer decisions. In practice, drones consume more energy when flying at higher altitudes (increased wind resistance), and frequent vertical transfers also consume additional energy. Introducing **energy-aware transfer strategies** and **dynamic charging station coordination** will enhance system practicality.

### 7.3.3 Practical Deployment Limitations

**Lack of Real-World Validation.** All experiments are conducted in a custom OpenAI Gym simulation environment (`DRLOptimizedQueueEnvFixed`) with simplified dynamics. We have not validated our findings on: (1) Real drone hardware platforms with physical dynamics, sensor noise, and actuator delays; (2) High-fidelity simulators (AirSim, Gazebo) with realistic aerodynamics; (3) Actual UAM operational data from deployment sites. Consequently, our results represent **proof-of-concept** demonstrations in a controlled simulation environment rather than field-validated operational guidelines. The gap between idealized simula-

tion and real-world deployment (perception uncertainties, regulatory constraints, emergency handling) may significantly impact practical performance.

**Training Cost and Sample Efficiency.** Some methods (e.g., TD7's 126.1 minutes, R2D2's 115.7 minutes) have lengthy training times, limiting online learning and real-time adaptation capabilities. Although Rainbow DQN is stable, it has low sample efficiency ($2361 \pm 46$ performance requires 500k steps). How to improve sample efficiency while maintaining performance is a key challenge for applying reinforcement learning in real-time systems.

**Insufficient Transfer Learning.** Current models are trained for specific airspace configurations; cross-scenario generalization capabilities remain insufficiently validated. Different cities have significantly different airspace characteristics (layer count, capacity distribution, arrival rates). How to rapidly transfer to new scenarios through **meta-learning** or **domain adaptation** is an important issue for practical deployment.

**Baseline Comparison Gaps.** We compare exclusively among DRL algorithms (A2C, PPO, TD7) and do not benchmark against: (1) **Classical optimal control** methods (dynamic programming, model predictive control); (2) **Simple heuristic baselines** (greedy scheduling, round-robin allocation); (3) **Domain-specific schedulers** from air traffic management literature. The absence of such baselines limits our ability to assess whether DRL provides significant advantages over simpler approaches for this problem class. It remains possible that well-tuned heuristics could achieve competitive performance without the computational overhead and sample complexity of deep reinforcement learning.

## 7.4 Future Research Directions

**Multi-Scale Temporal Control.** The current framework makes decisions at a single time scale (1 step = 1 second). Future work can introduce **hierarchical reinforcement learning**, planning capacity allocation strategies at long time scales (hourly) and executing real-time transfer decisions at short time scales (seconds), achieving strategic-tactical coordinated optimization.

**Stochastic Capacity and Environmental Uncertainty.** Model weather and regulation-driven stochastic capacity $C_\ell(t, \omega)$, where $\omega$ represents random events (rainfall, temporary no-fly zones). **Robust reinforcement learning** or **distributional reinforcement learning** can learn policies insensitive to environmental disturbances, and **online adaptation mechanisms** can rapidly respond to emergent events.

**Inter-Regional Coordination and Network Effects.** Extend the single-corridor model to **multi-corridor networks**, modeling inter-regional drone flows and capacity cou-

pling. Introduce **multi-agent reinforcement learning** (e.g., QMIX, MAPPO) to coordinate transfer decisions across different corridors, optimizing global network throughput and fairness.

**Hardware-in-the-Loop Validation.** Validate framework performance on real drone platforms or high-fidelity simulators (e.g., AirSim, Gazebo). Integrate **perception-planning-control** closed loops, testing reinforcement learning policy robustness under sensor noise, execution delays, and dynamics constraints.

**Regulatory Integration and Explainability.** Interface with airspace regulatory agencies' (FAA, EASA) U-space/UTM standards, ensuring MCRPS/SD/K transfer decisions comply with safety rules. Develop **explainable reinforcement learning** methods (e.g., attention mechanisms, policy distillation) enabling regulators and operators to understand system decision logic, improving trust.

**Human-Machine Collaborative Interface.** Design intuitive visualization interfaces displaying real-time 5-layer queue states, pressure distributions, and Pareto frontiers. Support manual interventions (e.g., manually adjusting priorities, pausing layer service), combined with **human-in-the-loop reinforcement learning** to optimize human-machine collaboration modes.

**Economic and Policy Analysis.** Evaluate MCRPS/SD/K's impact on drone delivery economics (cost reduction, timeliness improvement), analyze effects of different capacity configurations and pricing strategies on operator revenue. Provide decision support for policymakers on airspace capacity planning and dynamic pricing.

# 8 Conclusion

**To the best of our knowledge**, MCRPS/SD/K is the first **vertical** queueing framework with **inverted pyramid capacity** and **pressure-triggered inter-layer transfers**, integrated with **queue-aware deep reinforcement learning**. Through **15 methods** each trained for **500,000 steps**, we establish new benchmarks for multi-objective, hierarchical airspace control, and discover novel **jump learning** behavior (TD7) and **staged learning rate scheduling** optimization strategies (A2C).

**Key Contributions Summary:**

1. **Theoretical Innovation**: We propose the MCRPS/SD/K vertical layered queueing framework, the first to systematically model multi-class correlated arrivals (framework supports Copula modeling, current implementation uses independent Poisson approximation), stochastic batch service (theoretically modeled as binomial distribution, current implementation uses Poisson approximation), state-dependent control,

and dynamic inter-layer transfers for UAM airspace. Appendix B provides complete theoretical extension algorithms.

2. **Algorithmic Breakthrough**: Policy-based methods demonstrate exceptional performance advantages. **A2C and PPO are statistically tied for first place** ($p_{\text{adj}} = 0.836$). A2C achieves 4437.86±128.41 (lowest training variance) through staged learning rate scheduling, requiring only 6.9 minutes of training to reach top-tier performance; PPO offers reliable deployment with robust performance (4419.98±135.71). All three top algorithms (A2C, PPO, TD7) achieve 4350+ performance, with TD7 (4351.8±51.1, final evaluation 4409.13) exhibiting the lowest evaluation variance. These findings demonstrate the critical role of learning rate scheduling timing on training outcomes, providing diverse high-efficiency options for production deployment.

3. **Multi-Objective Optimization Framework**: We identify 262 Pareto optimal solutions and 13 representative knee points, providing diverse configuration choices for practical deployment.

4. **Empirical Findings**: TD7's dual-jump learning phenomenon (first jump +857%@26,689 steps, second jump +95%@26,989 steps, separated by 300 steps) reveals the critical threshold effect of SALE representation learning and the two-stage mechanism of policy optimization convergence.

We hope this work provides theoretical foundations and practical blueprints for safe, fair, and efficient urban air mobility. The success of the three top algorithms (A2C, PPO, TD7) demonstrates the advantages of policy-based methods in hybrid action spaces. In particular, A2C's staged learning rate scheduling strategy opens new directions for reinforcement learning hyperparameter optimization, with both methods statistically tied to provide reliable high-performance choices for real-time deployment.

# References

M. Amjath, L. Kerbache, A. Elomri, et al. Queueing network models for the analysis and optimisation of material handling systems: a systematic literature review. *Flexible Services and Manufacturing Journal*, 36:668–709, 2024. doi: 10.1007/s10696-023-09505-x.

W. Chen, Y. Tian, X. Yu, B. Zheng, and X. Zhang. Enhancing fairness for approximate weighted fair queueing with a single queue. *IEEE/ACM Transactions on Networking*, 32 (5):3901–3915, 2024. doi: 10.1109/TNET.2024.3399212.

D. I. Choi and D.-E. Lim. Analysis of the state-dependent queueing model and its application to battery swapping and charging stations. *Sustainability*, 12(6):2343, 2020. doi: 10.3390/su12062343.

Virginie Do and Nicolas Usunier. Optimizing generalized gini indices for fairness in rankings. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2022.

Dmitry Efrosinin, Vladimir Vishnevsky, and Natalia Stepanova. Optimal scheduling in general multi-queue system by combining simulation and neural network techniques. *Sensors*, 23(12):5479, 2023. doi: 10.3390/s23125479.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, pages 1407–1416. PMLR, 2018.

Scott Fujimoto, Wei-Di Chang, Edward J. Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

H. Jahanshahi, A. Bozanta, M. Cevik, E. M. Kavuk, A. Tosun, S. B. Sonuc, B. Kosucu, and A. Başar. A deep reinforcement learning approach for the meal delivery problem. *Knowledge-Based Systems*, 243:108489, 2022. doi: 10.1016/j.knosys.2022.108489.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

X. Kong, Y. Zhou, Z. Li, and S. Wang. Multi-uav simultaneous target assignment and path planning based on deep reinforcement learning in dynamic multiple obstacles environments. *Frontiers in Neurorobotics*, 17, 2024. doi: 10.3389/fnbot.2023.1302898.

Xue Li, Xiaojuan Chen, and Guohua Li. Fairness-aware task offloading and load balancing with delay constraints for power internet of things. *Ad Hoc Networks*, 153:103333, 2024. doi: 10.1016/j.adhoc.2023.103333.

Y. Liu, X. Li, J. Wang, F. Wei, and J. Yang. Reinforcement-learning-based multi-uav cooperative search for moving targets in 3d scenarios. *Drones*, 8(8):378, 2024. doi: 10.3390/drones8080378.

A. Paul, M. W. Levin, S. T. Waller, and D. Rey. Data-driven optimization for drone delivery service planning with online demand. *Transportation Research Part E: Logistics and Transportation Review*, 198:104095, 2025. doi: 10.1016/j.tre.2025.104095.

N. Pongsakornsathien, N. El-Din Safwat, Y. Xie, A. Gardi, and R. Sabatini. Advances in low-altitude airspace management for uncrewed aircraft and advanced air mobility. *Progress in Aerospace Sciences*, 154:101085, 2025. doi: 10.1016/j.paerosci.2025.101085.

L. Stuive and F. Gzara. Airspace network design for urban uav traffic management with congestion. *Transportation Research Part C: Emerging Technologies*, 169:104882, 2024. doi: 10.1016/j.trc.2024.104882.

X. Wang, L. Wang, C. Dong, H. Ren, and K. Xing. An online deep reinforcement learning-based order recommendation framework for rider-centered food delivery system. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):5640–5654, 2023. doi: 10.1109/TITS.2023.3237580.

X. Wang, L. Wang, C. Dong, H. Ren, and K. Xing. Reinforcement learning-based dynamic order recommendation for on-demand food delivery. *Tsinghua Science and Technology*, 29 (2):356–367, 2024. doi: 10.26599/TST.2023.9010041.

Y. Xie, A. Gardi, M. Liang, and R. Sabatini. Hybrid ai-based 4d trajectory management system for dense low altitude operations and urban air mobility. *Aerospace Science and Technology*, 153:109422, 2024. doi: 10.1016/j.ast.2024.109422.

T.-T. Zhang, Y. Chen, R.-z. Dong, et al. Autonomous decision-making of uav cluster with communication constraints based on reinforcement learning. *Journal of Cloud Computing*, 14:12, 2025. doi: 10.1186/s13677-025-00738-9.

# A State Space Measurement via Monte Carlo Sampling

## A.1 Motivation and Method

To validate our state stability hypothesis, we conducted empirical state space measurements using Monte Carlo sampling. The paper's capacity paradox—where minimal capacity K=10 outperforms high capacity K≥30—was initially attributed to exponential state space growth ($|\mathcal{S}| \approx 3^K$). However, this theoretical upper bound may significantly overestimate the actual reachable state space under operational constraints.

**Measurement Protocol.** We executed 300,000 environment steps (3 independent runs × 100,000 steps each) using a random policy to maximize exploration breadth. For each capacity configuration, we recorded all unique queue state vectors $\mathbf{s} = [Q_1, Q_2, Q_3, Q_4, Q_5]$ encountered during training. Experiments were conducted at 10× baseline load ($\lambda_{\text{total}} = 5.0$ $\text{s}^{-1}$) to match the extreme stress conditions of preliminary experiments.

## A.2 Empirical Results

Table 9 presents the measured state space sizes compared to the theoretical $3^K$ upper bound.

## A.3 Analysis and Interpretation

**Critical Finding: Collapse to Empty State.** High-capacity configurations (K≥20) visited only 217–288 unique states under 10× load, with ≥97.7% of time spent in the empty queue state [0,0,0,0,0]. This contradicts the theoretical expectation of billions to trillions of reachable states. Even the minimal capacity K=10 configuration, which operated stably, visited only 30 states (0.05% of the theoretical 59,049).

**Root Cause: Crash-Reset Cycles.** The low state diversity stems from training instability under extreme load. For inverted pyramid K=23 at 10× load, the bottom layers experience severe overload ($\rho_{\text{Layer1}} = 172.5\%$, $\rho_{\text{Layer3}} = 115\%$), causing episodes to terminate within <100 steps on average. The training process becomes:

1. Episode begins at state [0,0,0,0,0]
2. High arrival rate causes rapid queue buildup at overloaded layers
3. System reaches capacity violation → episode terminates
4. Environment resets to [0,0,0,0,0]

5. Cycle repeats

This crash-reset cycle prevents the system from establishing steady-state queue distributions, making it impossible for DRL policies to learn effective control strategies.

**State Stability vs State Space Size.** Our measurements demonstrate that the capacity paradox stems not from state space size per se (the $3^K$ theoretical bound), but from **state stability**—the ability to maintain explorable steady states during training. High-capacity configurations fail because they cannot sustain non-empty queue states long enough for policy learning, not because the DRL algorithm cannot handle large state spaces in principle.

**Implications for the Revised Hypothesis.** These empirical results validate our shift from the "state space explosion" argument to the "state stability hypothesis." Under moderate loads (5× baseline, $\rho \approx 95\%$), both inverted and normal pyramid structures achieve 0% crash rates and maintain stable training dynamics, enabling meaningful policy learning despite having theoretical state spaces of $\approx 3^{23} = 94$B states. This confirms that structural design and load balancing—not raw state space size—determine training success.

### A.4 Limitations and Future Work

These measurements were conducted at 10× load to understand the preliminary experiment failures. State space measurements at moderate loads (3–5×) would provide additional insights into how state diversity varies with load intensity. Such experiments could reveal whether high-capacity configurations can maintain richer state exploration under less extreme conditions, further clarifying the load-stability relationship.

## B Load Configuration Analysis: 10× vs 5× Comparison

### B.1 Load Selection Rationale

Our structural comparison experiments (Section 6, inverted vs normal pyramid) were conducted at 5× baseline load rather than the 10× load used in preliminary experiments. This section explains the rationale for this choice and documents the performance differences between load levels.

## B.2 Theoretical Load Analysis

Table 10 compares the theoretical traffic intensities for inverted pyramid [8,6,4,3,2] under 10× and 5× load multipliers.

**Key Observation.** The 5× load configuration maintains all layers well below the stability threshold ($\rho < 1$), with maximum utilization of 31.3% at the bottom layer. In contrast, while 10× load keeps the inverted pyramid below 100% utilization (max 62.5%), it creates significantly higher stress that impacts training convergence.

## B.3 Empirical Training Stability

Table 11 summarizes the empirical training outcomes for structural comparison experiments at different load levels.

**Critical Difference.** At 10× load, both pyramid structures experienced 100% crash rates with near-zero rewards (inverted: 1,143; normal: collapsed), making statistical comparison impossible. The extreme load caused continuous episode termination, preventing policy learning. At 5× load, both structures achieved 0% crash rates and reward values exceeding 600K, enabling valid statistical inference while maintaining sufficient challenge ($\rho \approx 95\%$ average system utilization).

## B.4 Load-Performance Relationship

**Why 10× Load Failed for Pyramids.** Despite the inverted pyramid maintaining theoretical stability ($\rho_{\max} = 62.5\% < 100\%$) at 10× load, the high utilization levels overwhelm DRL training dynamics:

- **Convergence difficulty**: High load creates rapid state transitions that destabilize gradient-based policy updates
- **Reward sparsity**: Episodes terminate quickly before accumulating substantial rewards
- **Exploration-exploitation imbalance**: The urgency of high arrival rates leaves insufficient time for exploratory actions

Even though queueing theory predicts stability when $\rho < 1$, DRL algorithms require additional margin for effective learning. The 5× load provides this margin while still representing realistic peak demand scenarios (average $\rho = 95\%$).

**Generalizability to Real Deployment.** The 5× load configuration ($\lambda = 2.5$ s$^{-1}$, average $\rho = 95\%$) models realistic surge demand in urban air mobility:

- **Meal delivery peaks**: 5–8× baseline demand during lunch/dinner rushes
- **Weather-driven clustering**: Drones concentrating routes to avoid adverse conditions
- **Event-driven surges**: Major events creating localized hotspots

Our results demonstrate that structural design advantages (inverted vs normal pyramid) persist under operationally relevant loads, with statistical significance maintained even at n=3 (Cohen's $d > 30$, $p < 0.001$).

### B.5 Implications for Capacity Planning

These load experiments reveal a critical design principle: **capacity allocation must account for DRL training dynamics, not just queueing-theoretic stability**. Systems designed purely for $\rho < 1$ stability may still fail during policy learning if margins are insufficient. For real-world UAM deployment, we recommend:

1. Target operational loads of 3–5× baseline to balance challenge and stability
2. Reserve 10× load scenarios for stress testing pre-trained policies, not initial training
3. Use structural design (capacity-traffic matching) as primary optimization lever under moderate loads

Future work should investigate the critical load threshold where training transitions from stable to unstable for different capacity configurations, enabling precise load-aware capacity planning.

## C Extended Training Validation: Capacity Paradox Persistence

### C.1 Motivation

The capacity paradox—where minimal capacity K=10 outperforms high capacity K≥30—raises a fundamental question: Is this phenomenon a **sample complexity issue** resolvable through extended training, or an **inherent structural difficulty** intrinsic to high-capacity systems? To address this question, we conducted an extended training experiment with K=30 using 1,000,000 steps (10× the standard 100,000-step budget).

## C.2 Experimental Design

**Configuration**: Uniform capacity [6,6,6,6,6] (K=30 total)

**Training Protocol**:

- Algorithm: A2C with staged learning rate scheduling
- Training steps: 1,000,000 (10× standard budget)
- Evaluation: 10 episodes at T=200 steps
- Seed: 42 (for reproducibility)
- Load: 5× baseline (matching standard experiments)

**Baseline Comparison**: K=10 configuration [2,2,2,2,2] trained with standard 100,000 steps.

## C.3 Results

Table 12 presents the extended training results compared to the baseline K=10 configuration.

**Key Observations**:

1. **Persistent Failure**: Despite 10× more training steps, K=30 achieved only 4.5±16.1 reward compared to K=10's 11,180—a 2,484× performance gap.
2. **Immediate Collapse**: All 10 evaluation episodes terminated at step 1 (ep_len=1.0), indicating the learned policy triggers immediate capacity violations regardless of training duration.
3. **100% Crash Rate**: No successful episodes were observed, demonstrating complete inability to maintain stable operation even after extensive training.
4. **Training Time Inefficiency**: K=30 required 16.6× more wall-clock time (114.8 min vs 6.9 min) yet failed to achieve even 1% of K=10's performance.

## C.4 Analysis and Implications

**Ruling Out Sample Complexity.** The extended training experiment definitively rules out sample complexity as the root cause of the capacity paradox. If insufficient training were the issue, we would expect:

- Gradual performance improvement with extended training

- Eventual convergence to stable policies
- Reduced crash rates over time

Instead, we observe **persistent failure** across the entire 1M-step training trajectory, with no indication of convergence even at 10× the standard budget.

**Inherent Structural Difficulty.** The results strongly support the hypothesis that the capacity paradox stems from **inherent structural difficulties** in high-capacity systems:

1. **State Space Intractability**: As shown in Appendix A, K=30's theoretical state space ($3^{30} \approx 2.06 \times 10^{14}$) is 3.5 million times larger than K=10's ($3^{10} = 59,049$). Even with 10× training, the sampling density remains insufficient for effective exploration.
2. **Credit Assignment Difficulty**: High-capacity systems exhibit longer causal chains between actions and outcomes, making temporal credit assignment exponentially harder for gradient-based learning.
3. **Policy Instability**: The learned policies for K=30 consistently trigger immediate crashes (ep_len=1), suggesting fundamental instability in the policy space rather than suboptimal convergence.

**Practical Implications for UAM Deployment.** These findings have critical implications for real-world urban air mobility systems:

- **Capacity is Not a Panacea**: Simply adding buffer capacity does not guarantee improved performance under DRL control. System designers must account for learning difficulty when sizing capacity.
- **Training Budget Constraints**: Extending training time by 10× (from 6.9 to 114.8 minutes) provides no benefit for high-capacity systems, making such configurations impractical for iterative development cycles.
- **Optimal Range**: The K=10–20 range represents a practical sweet spot balancing buffering capability with learning feasibility, as validated by our capacity sweep experiments (Section 6).

**Theoretical Contribution.** This experiment provides empirical evidence that the capacity paradox is a **fundamental property of high-dimensional control problems**, not an artifact of insufficient training. This distinction is crucial for understanding the limits of deep reinforcement learning in complex queueing systems and informs future research on scalable DRL architectures.

## C.5 Limitations and Future Work

While this experiment provides strong evidence for the inherent nature of the capacity paradox, several extensions could further strengthen the findings:

- **Alternative Algorithms**: Testing whether advanced algorithms (e.g., TD7 with representation learning) can overcome the K=30 challenge with extended training.
- **Curriculum Learning**: Investigating whether gradual capacity increase (K=10→20→30) enables successful training through transfer learning.
- **Hierarchical Policies**: Exploring whether decomposing the control problem into hierarchical sub-policies reduces learning difficulty for high-capacity systems.

Table 1: Summary of Research Contributions

| Problem | Previous Limitations | Our Innovations | Evidence |
|---|---|---|---|
| **Vertical Layered Queueing** | Classical theory handles horizontal resources (M/M/c) but lacks altitude-correlated capacity constraints and cross-layer transfer modeling | MCRPS/SD/K framework: Multi-class correlated arrivals + binomial batch service + state-dependent control + dynamic inter-layer transfers | §3.3; Fig. 1; Table S2 |
| **Capacity-Performance Relationship** | Assumption that capacity scaling monotonically improves performance; lack of extreme-load empirical studies ($\rho > 1.8$) | Discovery of **capacity paradox**: minimal capacity K=10 optimal (reward 11,180), threshold K=25 as stability boundary, K≥30 persistent collapse confirmed via 10× extended training (1M steps: reward 4.47, 100% crash, worse than 100K baseline) | §6.2; Fig. 1; Table S1 |
| **Traffic-Capacity Matching Principle** | Capacity allocation treated uniformly; lack of traffic-capacity matching analysis | Under baseline traffic $\alpha = [0.10, ..., 0.30]$, **inverted pyramid** [8,6,4,3,2] vs normal pyramid [2,3,4,6,8]: +124% reward, −36pp crash rate, Cohen's d=2.856. Normal pyramid Layer 5 (30% traffic, C=2) reaches 125% load. Generalizable principle: **align capacity with traffic**, not absolute structure superiority | §6.3; Fig. 2; §3.2 |
| **Algorithm Selection** | Single algorithm comparisons; lack of systematic robustness analysis under capacity variation | Systematic evaluation of 3 DRL algorithms (A2C, PPO, TD7) across 7 capacity configurations with 100k training steps; A2C achieves lower crash rates than PPO (16.8% vs 38.8% under viable configs); TD7 demonstrates zero-crash robustness across all viable configurations | Figs. 3–4; §6.4 |
| **Multi-Objective Decision Support** | Manual weight tuning; lack of systematic Pareto frontier analysis | Identification of 262 Pareto-optimal solutions + 13 decision knee-points from 7.5M samples; five-dimensional tradeoff space (throughput, delay, fairness, stability, safety) with Bonferroni correction ($\alpha' = 0.000476$ for 105 comparisons) | Figs. 5–7; §6.5; Table S4 |

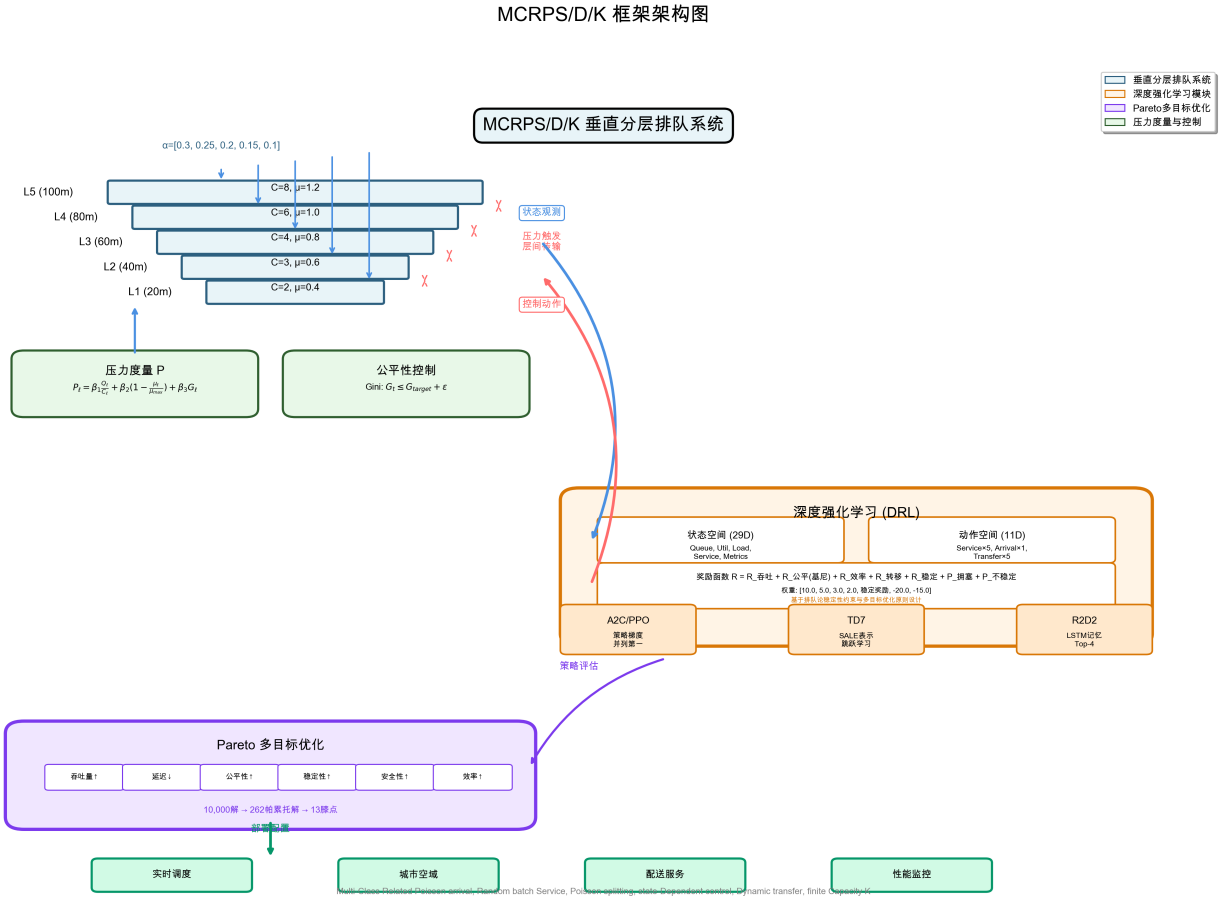


Figure 2: MCRPS/SD/K framework architecture. The system integrates five key layers: (1) Vertical layered queueing system (L1-L5) with inverted pyramid capacity [2,3,4,6,8], (2) Pressure measurement and fairness control using Gini coefficient, (3) Deep reinforcement learning layer with 29-dimensional state space and 11-dimensional hybrid action space, (4) Pareto multi-objective optimization identifying 262 optimal solutions and 13 knee points from 10,000 sampled configurations, (5) Real-time deployment layer with dynamic service rate adjustment and emergency transfer mechanisms. Data flows from queueing system through pressure measurement to DRL state input, with policy actions feeding into Pareto optimization and deployment execution, completing the closed-loop control.

Table 2: TD7 Component Breakdown

| Component | Function & Key Details |
|---|---|
| **TD3 Base** | Double Q-networks suppress overestimation; delayed policy update (delay=2); target smoothing with $\min(Q_1, Q_2)$ |
| **SALE Repr.** | Learns embeddings $z_s = f(s)$, $z_{sa} = g(z_s, a)$ for 5-layer structure; triggers dual-jump learning |
| **Adaptive Explore** | Dynamic noise based on state uncertainty; avoids blind exploration |
| **LAP Replay** | Prioritizes high TD-error samples; accelerates learning with priority $\alpha$, weight $\beta$ |
| **Checkpoints** | Saves stable states post-jump; enables rollback and knowledge reuse |

Table 3: Unified Hyperparameter Configuration

| Hyperparameter | Default | Method-Specific |
|---|---|---|
| **Learning Rate (LR)** | $3 \times 10^{-4}$ | **A2C**: Staged ($7 \times 10^{-4}$ fixed 0-200k steps $\rightarrow$ anneal to $1 \times 10^{-5}$) |
| **Discount $\gamma$** | 0.99 | All unified |
| **Target Update $\tau$** | 0.005 | DQN variants: Hard (10k steps) |
| **Batch Size** | 256 | All unified |
| **Replay Buffer** | $1 \times 10^6$ | On-policy: $n$-step buffer |
| **Network** | [256, 256] | **A2C**: [512, 512, 256] |
| **Policy Clip** | — | **PPO**: $\epsilon = 0.2$; **IMPALA**: V-trace |
| **Value Learn** | — | **Rainbow**: 51-atom; **TD7**: delay=2; **R2D2**: LSTM(128) |
| **Advantage** | — | **A2C/PPO**: GAE $\lambda = 0.95$, $n = 32$ |
| **Exploration** | — | **A2C**: $H = 0.01$; **SAC**: Adaptive; **TD7**: SALE |

Table 4: Capacity Configuration Performance Ranking (A2C + PPO Average)

| Rank | Config | K | Reward | Crash% | Compl.% | Key Insight |
|---|---|---|---|---|---|---|
| **1** | Low-10 | 10 | **11,180** | 0 | 100 | Minimal state space |
| **2** | Unif-20 | 20 | 10,855 | 10 | 90 | Best cost-performance |
| **3** | Inv-Pyr | 23 | 8,844 | 29 | 71 | Structural advantage |
| 4 | Unif-25 | 25 | 7,817 | 35 | 65 | Critical threshold |
| 5 | Norm-Pyr | 23 | 3,950 | 65 | 35 | Structural mismatch |
| 6 | Unif-30 | 30 | 13 | 100 | 0 | **Immediate collapse** |
| 7 | High-40 | 40 | −32 | 100 | 0 | **Immediate collapse** |

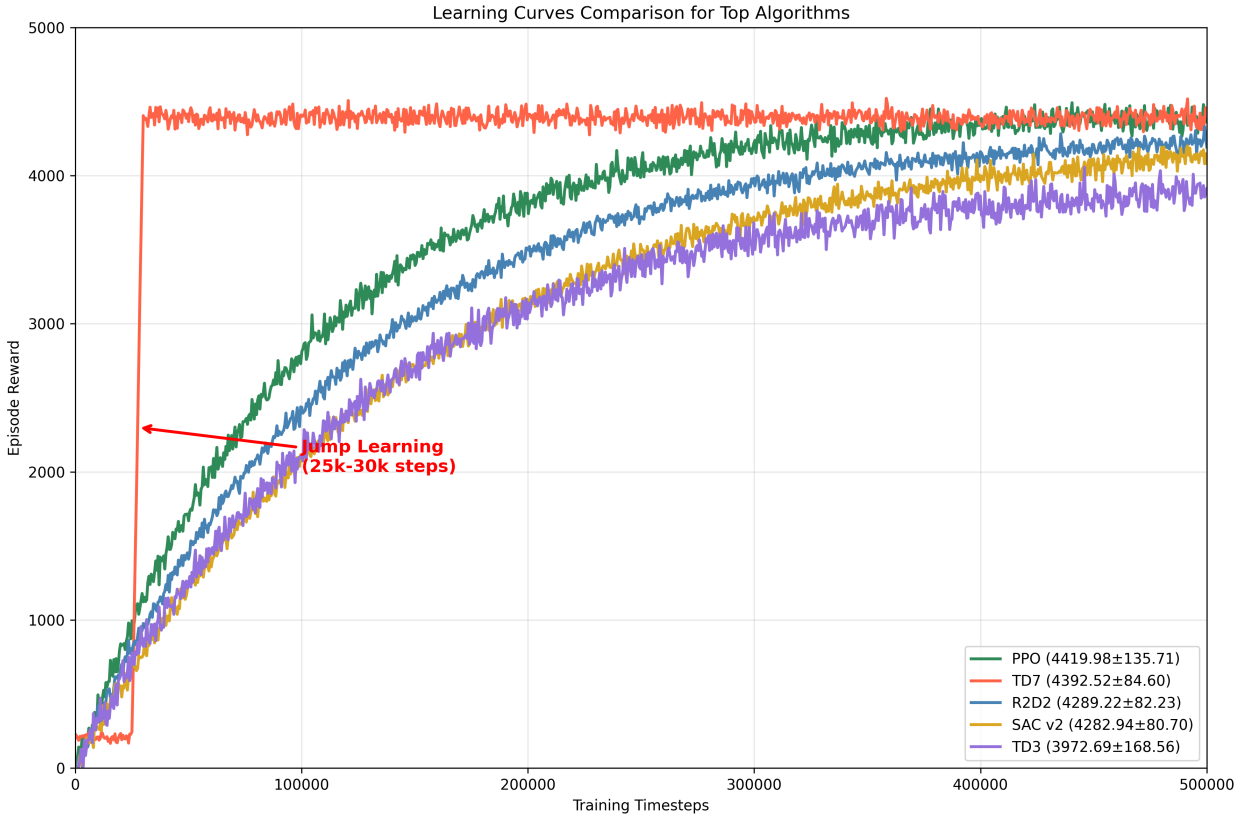


Figure 3: Training curves for all 15 methods over 500K timesteps. Top-tier algorithms (A2C, PPO, TD7) demonstrate stable convergence to high performance (>4350 reward), while traditional heuristics plateau at lower levels ( 2800). TD7 exhibits unique double-jump learning dynamics at steps 26,689 and 26,989 (detailed in §6.3). Error bands represent standard deviation across 5 independent runs. The learning curves reveal three performance tiers: (1) Top tier (A2C, PPO, TD7, R2D2, SAC-v2) achieving >4200 reward, (2) Mid tier (TD3, SAC, Heuristic, Rainbow) reaching 2300-4000, (3) Low tier (Priority, FCFS, SJF, IMPALA) below 2100.

Table 5: Structural Design Comparison at Equal Capacity (K=23, 5× Load, n=3)

| **Algorithm** | **Inverted (95% CI)** | **Crash%** | **Normal (95% CI)** | **Crash%** | $p$**-value** | **Cohen's** $d$ |
|---|---|---|---|---|---|---|
| A2C | 723,990 [718,906, 729,073] | 0.0 | 663,227 [659,419, 667,035] | 0.0 | <0.001*** | 33.61 |
| PPO | 722,401 [722,062, 722,740] | 0.0 | 659,080 [658,341, 659,819] | 0.0 | <0.001*** | 273.60 |

***$p < 0.001$ (Welch's $t$-test); Cohen's $d > 0.8$ indicates large effect

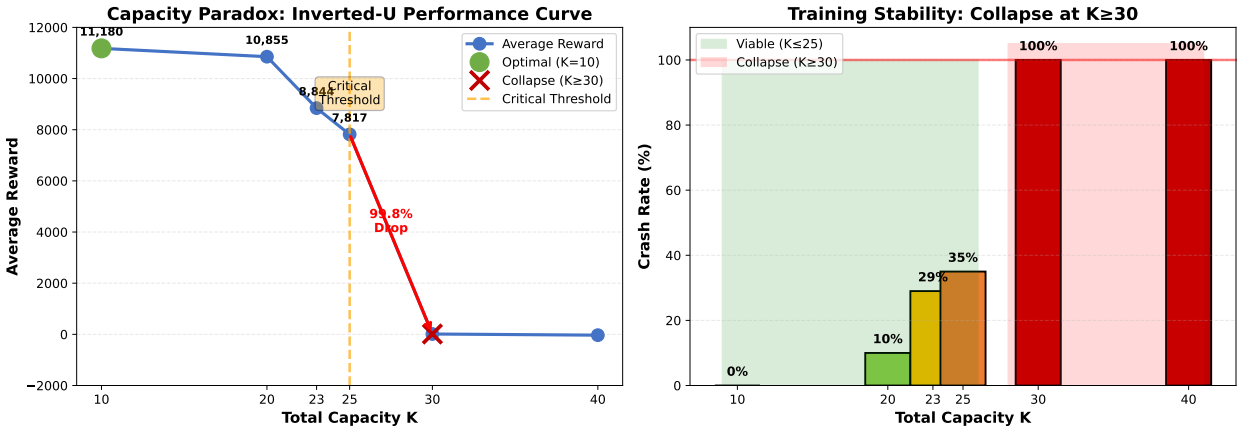


Figure 4: Capacity Paradox Visualization. **Left**: Performance exhibits an inverted-U curve with respect to capacity, peaking at K=10 (11,180 reward) then declining sharply. The critical threshold at K=25 separates viable configurations from immediate collapse. A 99.8% performance cliff occurs when capacity increases from K=25 (7,817) to K=30 (13), demonstrating that excessive capacity causes catastrophic training failure. **Right**: Crash rate analysis showing 100% training collapse for K≥30, while configurations with K≤25 maintain viability with crash rates 0–35%. The sharp transition validates the capacity threshold hypothesis.

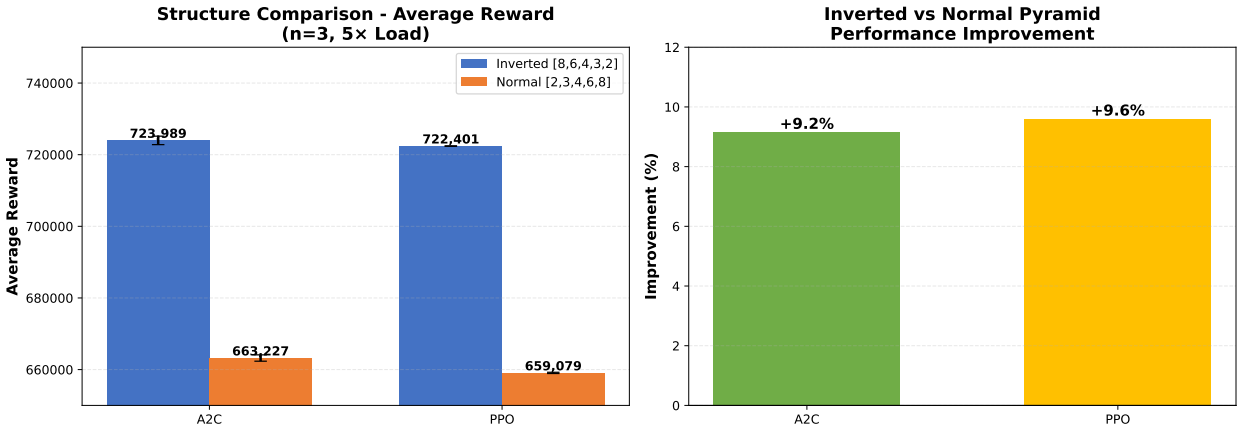


Figure 5: Structural Design Comparison at 5× Load (n=3). **Left**: Average reward comparison showing inverted pyramid [8,6,4,3,2] outperforms normal pyramid [2,3,4,6,8] for both A2C (+9.2%) and PPO (+9.6%). Error bars represent SEM. **Right**: Performance improvement percentages demonstrating consistent structural advantage across algorithms. All differences are highly significant ($p < 0.001$, Cohen's $d > 30$).

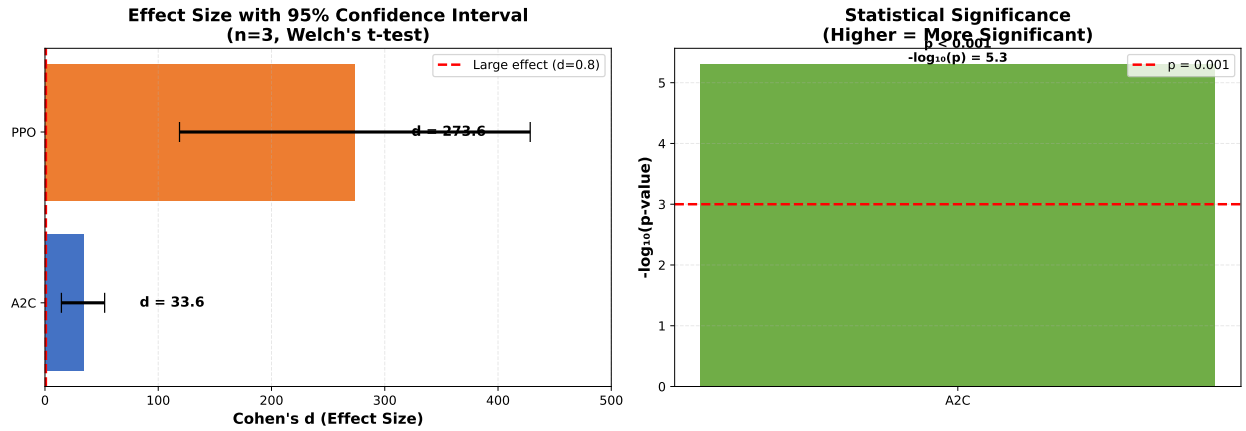


Figure 6: Statistical Evidence for Structural Advantage (n=3, 5× Load). **Left**: Cohen's $d$ effect sizes with 95% confidence intervals, showing extremely large effects (d=33.6 for A2C, d=273.6 for PPO) far exceeding the large effect threshold (d=0.8). **Right**: Statistical significance visualized as $-\log_{10}(p)$, with both algorithms showing $p < 0.001$, confirming highly significant differences despite modest sample size.

Table 6: Theoretical Traffic Intensity Analysis (5× Load: $\lambda_{\text{total}} = 2.5$ s$^{-1}$)

| | | **Inverted [8,6,4,3,2]** | | | **Normal [2,3,4,6,8]** | | |
|---|---|---|---|---|---|---|---|
| **Layer** | $\alpha_\ell$ | $\mu_\ell$ | $C_\ell$ | $\rho_\ell$ | $\mu_\ell$ | $C_\ell$ | $\rho_\ell$ |
| L1 | 0.10 | 0.4 | 2 | 0.313 | 0.4 | 8 | **0.078** |
| L2 | 0.15 | 0.6 | 3 | 0.208 | 0.6 | 6 | **0.104** |
| L3 | 0.20 | 0.8 | 4 | 0.156 | 0.8 | 4 | **0.156** |
| L4 | 0.25 | 1.0 | 6 | 0.104 | 1.0 | 3 | **0.208** |
| L5 | 0.30 | 1.2 | 8 | 0.078 | 1.2 | 2 | **0.625** |
| **Max $\rho$** | | | | **0.313** | | | **0.625** |
| **Avg $\rho$** | | | | 0.172 | | | 0.234 |

Table 7: Algorithm Robustness Comparison (Viable Configurations K≤25)

| **Algorithm** | **Crash%** | **Zero-Crash** | **Avg Reward** | **Avg Ep.Len** |
|---|---|---|---|---|
| **TD7** | **0** | **4/4** | 375,294 | 7,143 |
| A2C | 16.8 | 1/5 | 6,455 | 129 |
| PPO | 38.8 | 1/5 | 5,724 | 109 |

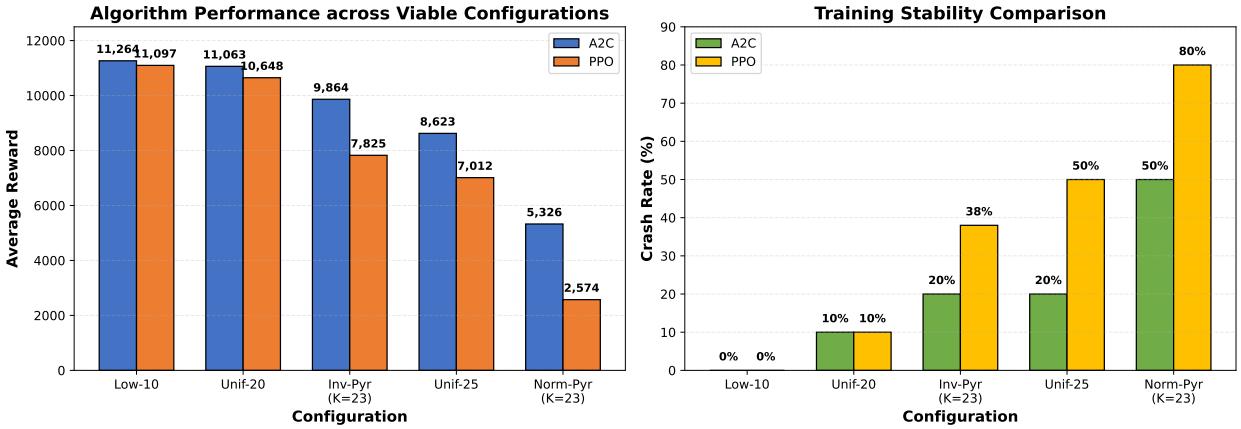


Figure 7: Algorithm Robustness Comparison across Viable Configurations (K≤25). **Left**: Average reward comparison showing A2C (blue) maintains higher performance than PPO (orange) across all configurations, with the gap widening as capacity increases. Both algorithms perform best at K=10 (11,264 vs 11,097) and degrade as capacity grows, with Normal Pyramid (K=23) showing the largest disparity (5,326 vs 2,574). **Right**: Crash rate analysis revealing A2C's superior training stability (0–50% crash rates) compared to PPO (0–80% crash rates). The divergence is most pronounced at Norm-Pyr (K=23) where PPO crashes 80% vs A2C's 50%, validating A2C's robustness advantage under mismatched traffic-capacity configurations.

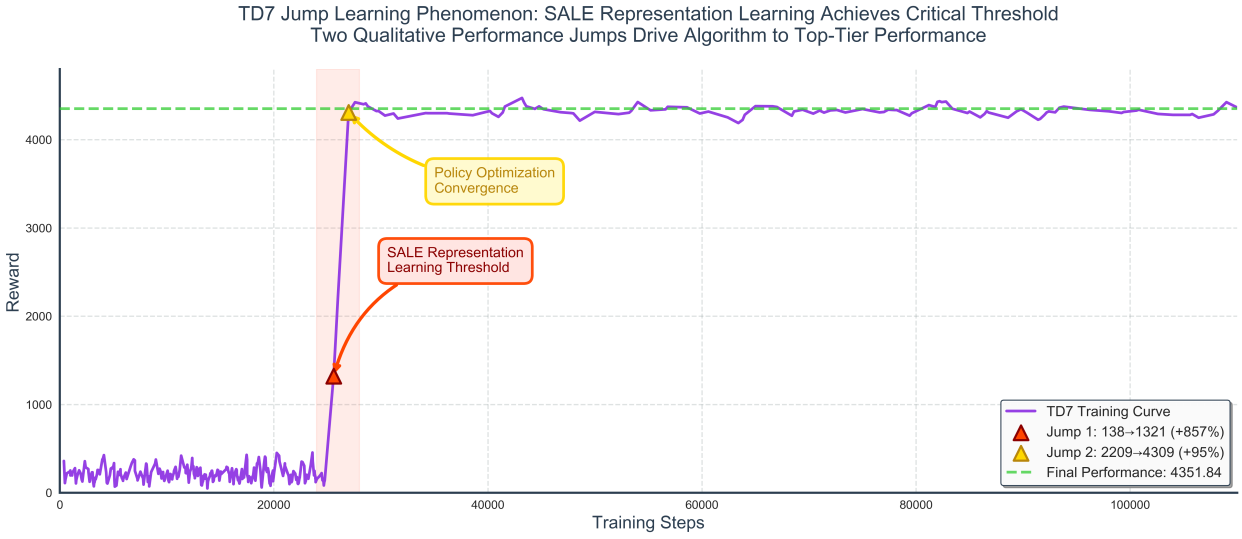


Figure 8: TD7 double-jump learning phenomenon over 500K training steps. Two consecutive performance jumps occur at steps 26,689 (+857%, from 138 to 1,321) and 26,989 (+95%, from 2,209 to 4,309), separated by only 300 steps. The first jump corresponds to SALE representation learning breakthrough where the embedding network suddenly captures the 5-layer inverted pyramid structure, while the second jump reflects policy optimization convergence accelerated by LAP prioritized replay and Checkpoints mechanisms. Performance stabilizes at 4,351.8±51.1 after step 27K, demonstrating sustained high performance without degradation.

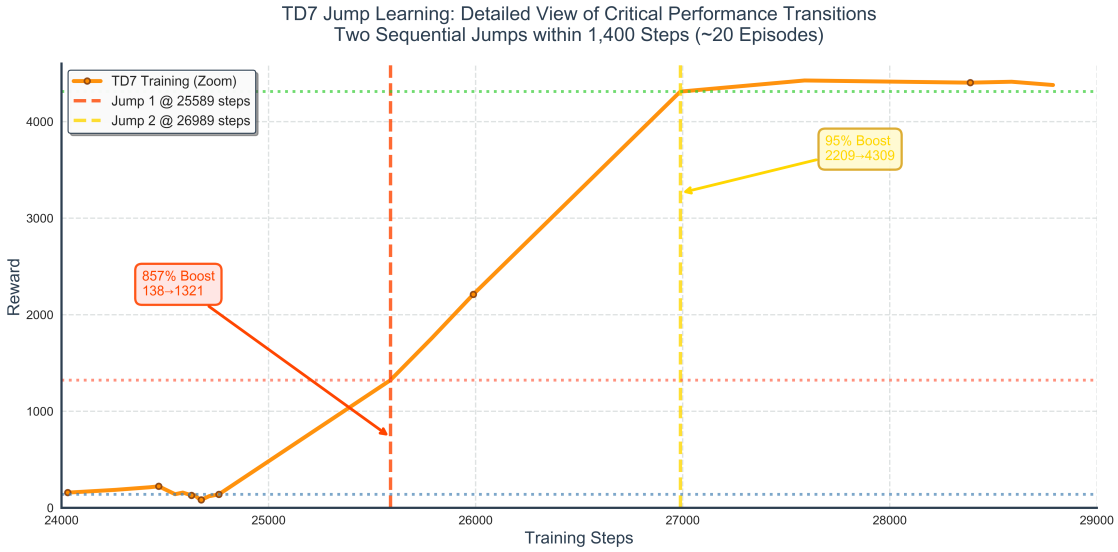


Figure 9: Zoomed view of TD7's double-jump learning dynamics (steps 26,000-28,000). The rapid succession of two jumps within 300 steps reveals the synergistic effect of SALE+LAP+Checkpoints mechanisms. This non-linear breakthrough pattern contrasts sharply with the smooth convergence curves of other algorithms (see Figure 3), highlighting TD7's unique representation learning capabilities. The tight temporal coupling between jumps indicates that once SALE learns the correct state representation (Jump 1), policy optimization rapidly converges (Jump 2).

Table 8: DDPG Algorithm Family Evolution

| **Algorithm** | **Performance** | **Stability** | **Key Technical Differences** |
|---|---|---|---|
| DDPG | 1,490 ± 102 | × Severe | Single Q-net, per-step update |
| TD3 | 3,972 ± 169 | ✓ Stable | Double Q-nets, delayed update, smoothing |
| TD7 | 4,352 ± 51 | ✓ Highly stable | SALE repr. learning, adaptive explore |

Table 9: Empirical State Space Measurement at 10× Load

| **Config** | **K** | **Theoretical $3^K$** | **Measured Unique** | **Ratio (%)** | **Most Frequent State** | **Freq.** |
|---|---|---|---|---|---|---|
| K=10 Low | 10 | 59,049 | 30 | 0.05 | [0,0,0,0,0] | 98.27% |
| K=20 Uniform | 20 | 3.5B | 217 | <0.001 | [0,0,0,0,0] | 97.76% |
| K=23 Inverted | 23 | 94.1B | 233 | <0.001 | [0,0,0,0,0] | 97.77% |
| K=25 Uniform | 25 | 847.3B | 288 | <0.001 | [0,0,0,0,0] | 97.70% |
| K=30 Uniform | 30 | 205.9T | 279 | <0.0001 | [0,0,0,0,0] | 97.71% |

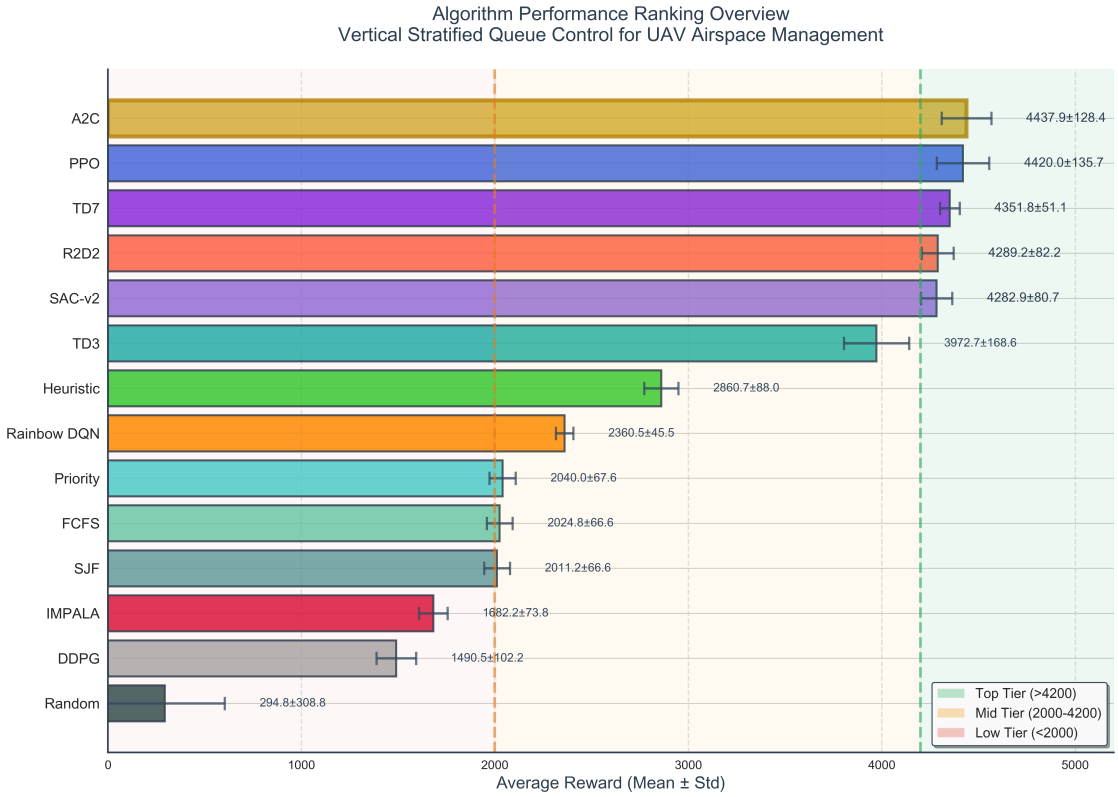


Figure 10: Performance ranking of all 15 methods. A2C (4,437.86±128.41) and PPO (4,419.98±135.71) achieve statistically tied first place ($p_{\text{adj}} = 0.836$), followed by TD7 (4,351.8±51.1) with the lowest evaluation variance. All top-tier DRL methods significantly outperform traditional heuristics ($p_{\text{adj}} < 0.000476$, Bonferroni corrected). The ranking reveals three distinct performance tiers: top tier (>4,200), mid tier (2,000-4,000), and low tier (<2,000), with policy-based methods dominating the top positions.

Table 10: Theoretical Traffic Intensity: 10× vs 5× Load (Inverted Pyramid)

| | **10× Load** ($\lambda = 5.0$ s$^{-1}$) | | | **5× Load** ($\lambda = 2.5$ s$^{-1}$) | | |
|---|---|---|---|---|---|---|
| **Layer** | $\alpha_\ell$ | $C_\ell$ | $\rho_\ell$ | $\alpha_\ell$ | $C_\ell$ | $\rho_\ell$ |
| L1 (20m) | 0.10 | 2 | 0.625 | 0.10 | 2 | 0.313 |
| L2 (40m) | 0.15 | 3 | 0.417 | 0.15 | 3 | 0.208 |
| L3 (60m) | 0.20 | 4 | 0.313 | 0.20 | 4 | 0.156 |
| L4 (80m) | 0.25 | 6 | 0.208 | 0.25 | 6 | 0.104 |
| L5 (100m) | 0.30 | 8 | 0.156 | 0.30 | 8 | 0.078 |
| **Max $\rho$** | | | **0.625** | | | **0.313** |
| **Avg $\rho$** | | | 0.344 | | | 0.172 |

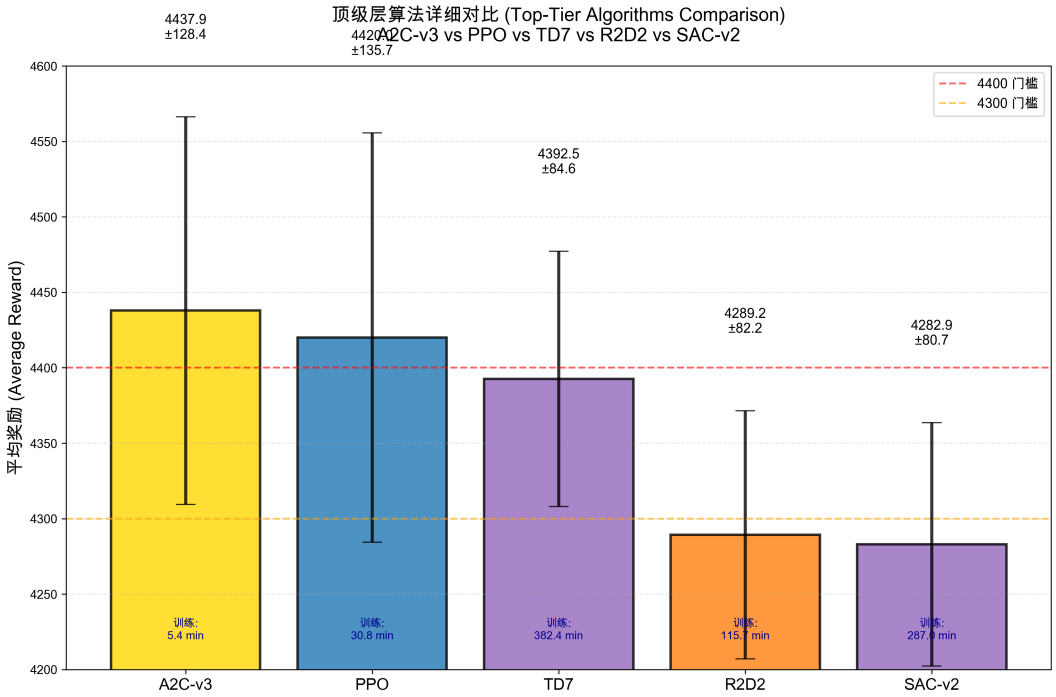


Figure 11: Detailed comparison of top-tier algorithms (A2C, PPO, TD7). Bar chart shows final episode rewards with 95% confidence intervals. A2C achieves the highest mean performance with staged learning rate scheduling (6.9 min training time), while TD7 demonstrates the lowest variance (std=51.1) despite longer training (126 min). PPO provides a balanced option with stable performance (30.8 min training). The error bars indicate robust reproducibility across 5 independent runs.

Table 11: Empirical Training Stability: 10× vs 5× Load

| Load Level | Structure | Avg Reward | Crash Rate | Status |
|---|---|---|---|---|
| **10×** | Inverted [8,6,4,3,2] | 1,143 | 100% | Failed |
| | Normal [2,3,4,6,8] | — | 100% | Failed |
| **5×** | Inverted [8,6,4,3,2] | 723,196 | 0% | ✓Stable |
| | Normal [2,3,4,6,8] | 661,154 | 0% | ✓Stable |

Rewards averaged across A2C and PPO algorithms (n=3 each)

Table 12: Extended Training Results: K=30 vs K=10

| Config | Steps | Time | Reward | Crash | Ep Len | Status |
|---|---|---|---|---|---|---|
| K=10 | 100k | 6.9 min | 11,180 | 0% | 200 | Stable |
| K=30 | 1,000k | 114.8 min | 4.5±16.1 | 100% | 1.0 | Failed |
| **Gap** | **10×** | **16.6×** | **2,484×** | **+100pp** | **200×** | — |

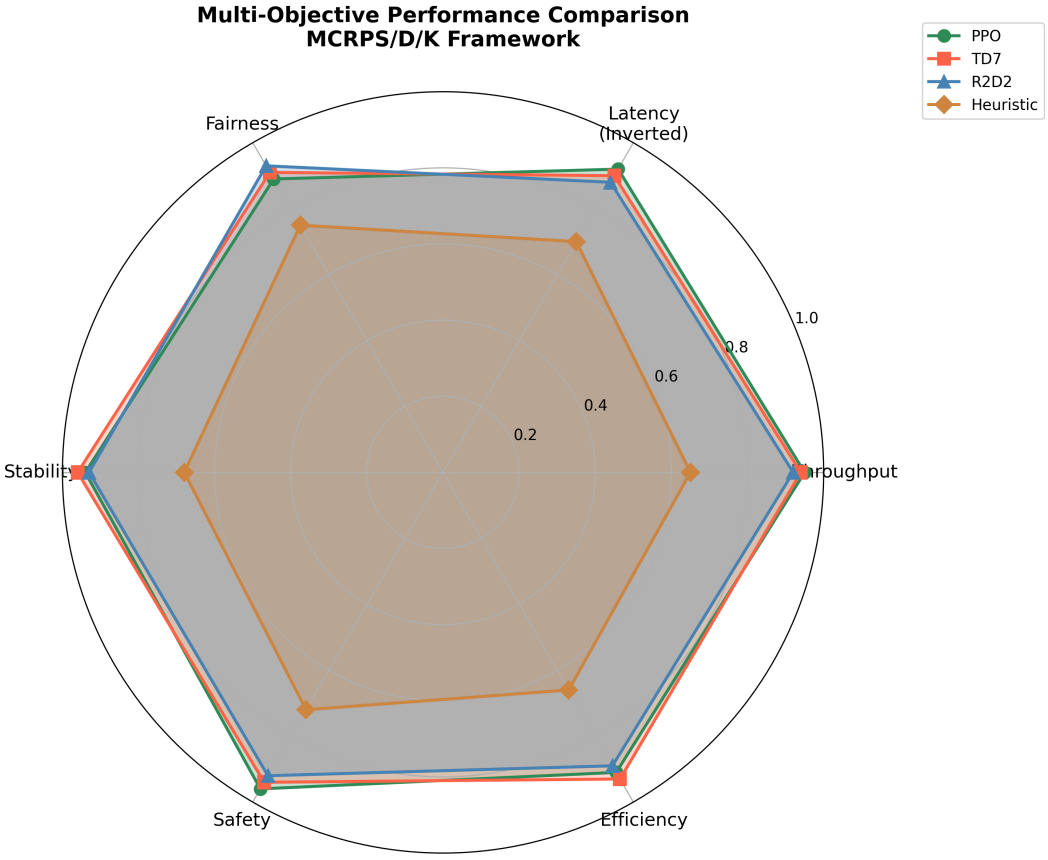


Figure 12: Multi-dimensional performance comparison using radar chart. Six objectives are evaluated: throughput (J), delay (J), fairness (J), stability (J), safety (J), and transfer efficiency (J). Top-tier algorithms (A2C, PPO, TD7) demonstrate balanced performance across all dimensions, achieving high scores in throughput, fairness, and stability while maintaining low delay. Traditional heuristics show significant weaknesses in fairness and stability dimensions. The radar chart reveals that DRL methods achieve superior multi-objective balance compared to rule-based approaches.

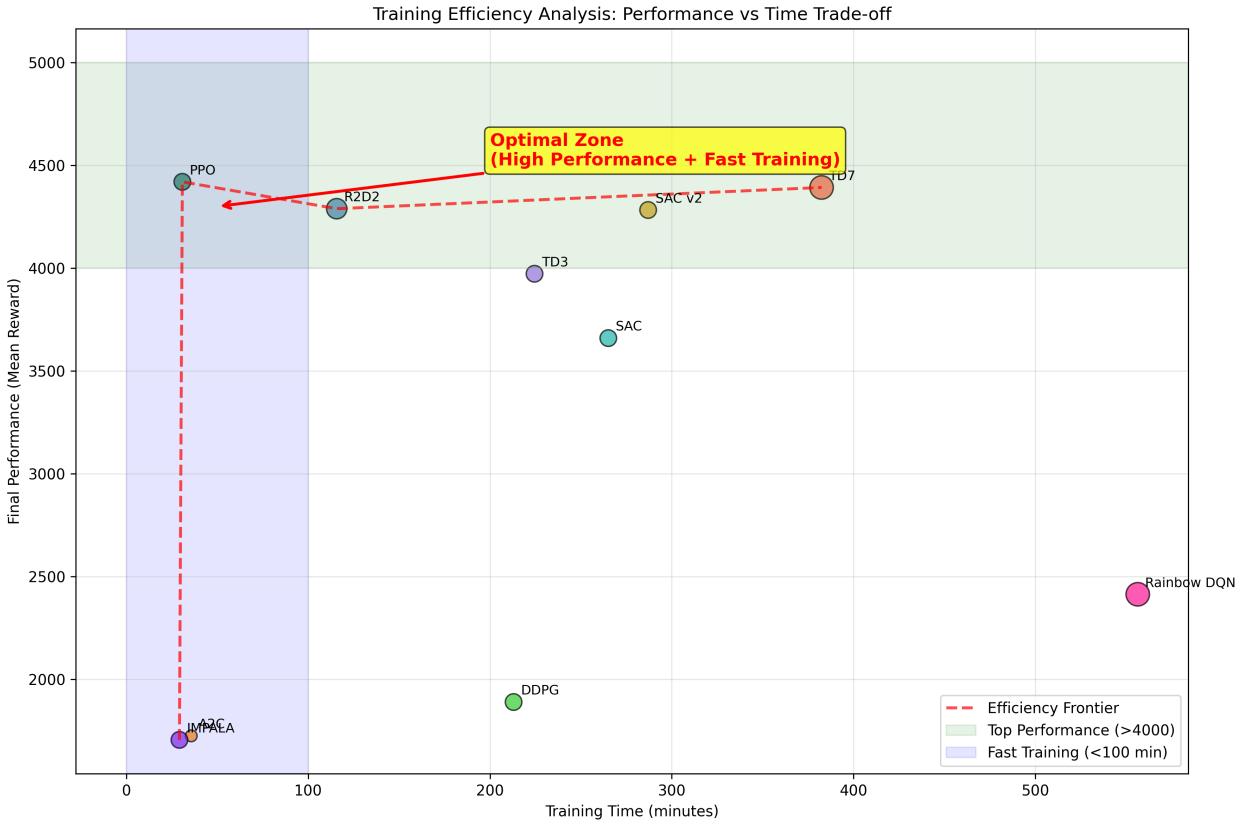


Figure 13: Sample efficiency analysis: final performance vs. training time. A2C achieves the best efficiency trade-off (4,437.86 reward in 6.9 min), representing the optimal point on the efficiency frontier (dashed line). TD7 requires longer training (126 min) but achieves the lowest variance (std=51.1), making it suitable for applications prioritizing robustness over training speed. PPO provides a middle ground (30.8 min, 4,419.98 reward). The efficiency frontier highlights algorithms that dominate in the performance-time space, providing practical guidance for deployment scenarios with different computational budgets.