aerospace

MDPI

Article

Cross-Domain Deep Reinforcement Learning for Real-Time Resource Allocation in Transportation Hubs: From Airport Gates to Seaport Berths

Zihao Zhang [1], Qingwei Zhong [1,*], Weijun Pan [2], Yi Ai [1] and Qian Wang [1]

[1] College of Air Traffic Management, Civil Aviation Flight University of China, Deyang 618311, China; zhangzihao@cafuc.edu.cn (Z.Z.); aiyi_cafuc@163.com (Y.A.); 15686352542@163.com (Q.W.)
[2] CAAC Academy of Flight Technology and Safety, Civil Aviation Flight University of China, Deyang 618311, China; wjpan@cafuc.edu.cn
* Correspondence: qingweizhong@cafuc.edu.cn

Abstract

Efficient resource allocation is critical for transportation hub operations, yet current scheduling systems require substantial domain-specific customization when deployed across different facilities. This paper presents a domain-adaptive deep reinforcement learning (DADRL) framework that learns transferable optimization policies for dynamic resource allocation across structurally similar transportation scheduling problems. The framework integrates dual-level heterogeneous graph attention networks for separating constraint topology from domain-specific features, hypergraph-based constraint modeling for capturing high-order dependencies, and hierarchical policy decomposition that reduces computational complexity from $O(mnT)$ to $O(m+n+T)$. Evaluated on realistic simulators modeling airport gate assignment (Singapore Changi: 50 gates, 300–400 daily flights) and seaport berth allocation (Singapore Port: 40 berths, 80–120 daily vessels), DADRL achieves 87.3% resource utilization in airport operations and 86.3% in port operations, outperforming commercial solvers under strict real-time constraints (Gurobi-MIP with 300 s time limit: 85.1%) while operating 270 times faster (1.1 s versus 298 s per instance). Given unlimited time, Gurobi achieves provably optimal solutions, but DADRL reaches 98.7% of this optimum in 1.1 s, making it suitable for time-critical operational scenarios where exact solvers are computationally infeasible. Critically, policies trained exclusively on airport scenarios retain 92.4% performance when applied to ports without retraining, requiring only 800 adaptation steps compared to 13,200 for domain-specific training. The framework maintains 86.2% performance under operational disruptions and scales to problems three times larger than training instances with only 7% degradation. These results demonstrate that learned optimization principles can generalize across transportation scheduling problems sharing common constraint structures, enabling rapid deployment of AI-based scheduling systems across multi-modal transportation networks with minimal customization and reduced implementation costs.

Keywords: deep reinforcement learning; meta-learning; combinatorial optimization; cross-domain generalization; dynamic scheduling

Check for updates

Academic Editor: Joost Ellerbroek

Received: 24 November 2025
Revised: 28 December 2025
Accepted: 8 January 2026
Published: 22 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Combinatorial optimization in dynamic scheduling represents one of the most challenging problems at the intersection of operations research and artificial intelligence.

From airport gate assignment to port berth allocation, from cloud computing resource management to smart grid energy distribution, such problems are ubiquitous in the operation of critical infrastructure, directly impacting global economic efficiency and the daily lives of billions of people. These problems share common characteristics: exponentially growing decision spaces, complex and diverse constraint conditions, and the requirement for near-optimal real-time decisions in highly uncertain dynamic environments. Despite significant advances in exact algorithms and heuristic methods over half a century of operations research, existing approaches heavily rely on domain expertise, struggle to transfer across different application scenarios, and suffer from severe computational inefficiency when facing large-scale real-time problems.

Consider airport gate assignment and port berth allocation as exemplars—two seemingly disparate problems that actually share identical mathematical structures. Both are NP-hard dynamic resource allocation problems requiring optimal assignment of limited resources to stochastically arriving demand entities under temporal window constraints, and both face challenges of multi-objective trade-offs and cascading effects. Yet current solutions rely almost entirely on domain-specific approaches. Airport operators employ specialized gate assignment systems, port authorities use independently developed berth scheduling software, and each system demands substantial human effort for algorithm design, parameter tuning, and maintenance updates.

Modern metaheuristics such as Adaptive Large Neighborhood Search (ALNS) [1] and genetic algorithms [2] demonstrate greater flexibility than hard-coded dispatching rules, adapting to different constraint configurations through algorithmic parameters and adaptive operator selection. However, even these sophisticated methods face significant limitations in cross-domain transfer. Transferring a well-tuned ALNS configuration from airport gate assignment to port berth allocation—despite their shared mathematical structure—typically requires redesigning neighborhood operators (e.g., replacing "swap adjacent gates" with "shift berth assignments"), re-specifying destruction and repair heuristics to respect domain-specific constraints (tidal windows versus aircraft turnaround requirements), and recalibrating acceptance criteria and adaptive weights from scratch. Each adaptation demands substantial domain expertise and empirical tuning. More critically, when business models evolve or unexpected disruptions occur, these systems require expert intervention for readjustment. This limited cross-domain adaptability, combined with the need for problem-specific customization, constrains the scalable deployment of optimization technologies across operationally diverse but structurally similar scheduling domains.

Deep reinforcement learning (DRL) has emerged as a promising paradigm to address this impasse. Through end-to-end learning, DRL can directly acquire decision policies from data without manual rule engineering. Recent years have witnessed breakthrough successes of DRL in domains such as gaming and robotic control, with nascent applications for combinatorial optimization problems. However, existing DRL methods face three fundamental challenges when tackling dynamic scheduling problems:

1. Insufficient generalization—most methods train for single problems and fail to transfer to new scenarios.
2. Poor sample efficiency—the discrete high-dimensional action spaces in combinatorial optimization render exploration extremely difficult, requiring massive data for convergence.
3. Inadequate constraint handling—real-world scheduling involves both hard constraints (e.g., physical capacity limits) and soft constraints (e.g., service quality requirements), and existing methods struggle to simultaneously guarantee feasibility and optimization performance.

https://doi.org/10.3390/aerospace13010108

This work introduces a cross-domain deep reinforcement learning framework that addresses these limitations through explicit modeling of structural invariances. Our core innovations include the following: a universal state representation mechanism that uniformly maps dynamic scheduling problems from different domains into an isomorphic Markov Decision Process (MDP) space; a hierarchical attention network architecture that adaptively captures problem-specific structural features and constraint relationships; a constraint-aware reward shaping method that transforms hard constraints into differentiable penalty signals, unifying feasibility guarantee with performance optimization; and a meta-reinforcement learning training strategy that enables the model to learn universal solution patterns from diverse problem instances, dramatically improving sample efficiency and zero-shot transfer capability.

We validated our framework's effectiveness on two real-world scenarios: airport gate assignment and port berth allocation. Experimental results demonstrate that using only 5% of domain knowledge, our method achieves performance comparable to expert systems on gate assignment (optimization gap <3%) and outperforms state-of-the-art heuristics by 12–18% on berth allocation. More importantly, models trained on airport scenarios directly applied to port scenarios achieve 92% performance retention without retraining, realizing truly cross-domain transfer in combinatorial optimization for the first time. Furthermore, our approach maintains stable performance when handling problem instances scaled to three times the training size, exhibiting exceptional generalization capability. Through ablation studies and visualization analysis, we reveal that the model automatically learns abstract patterns of classical operations research heuristics (such as First-Fit and Best-Fit) and develops superior decision strategies built upon this foundation.

This work not only provides a practical universal solver for dynamic scheduling problems but, more importantly, demonstrates the potential for deep integration between deep learning and operations research: by learning the intrinsic structure of problems rather than surface features, AI systems can transfer knowledge across domains like human experts. This paradigm shift holds significant implications for accelerating AI deployment at scale in industrial optimization. Methodologically, our domain-agnostic representation learning and constraint-aware training strategies offer new insights for addressing other NP-hard combinatorial optimization problems, including vehicle routing, production scheduling, and task assignment. As global infrastructure undergoes intelligent transformation, autonomous learning, rapid deployment, and continuously optimizing universal decision systems will become the core competitive advantage of next-generation intelligent operations.

2. Related Works

2.1. Deep Reinforcement Learning for Combinatorial Optimization

The application of deep reinforcement learning (DRL) to combinatorial optimization (CO) problems has emerged as a promising paradigm over the past decade. Pioneering work by Vinyals et al. [3] introduced Pointer Networks, demonstrating that sequence-to-sequence models with attention mechanisms could learn to solve routing problems such as the Traveling Salesman Problem (TSP). Building upon this foundation, Kool et al. [4] proposed the Attention Model, which employs graph attention networks to encode problem instances and generate solutions autoregressively, achieving competitive performance on various vehicle routing problems. Subsequent research has explored diverse neural architectures for CO, including graph convolutional networks for maximum cut problems [5], capsule networks for job shop scheduling [6], and transformer-based models for bin packing [7].

Despite these advances, existing DRL approaches for CO suffer from several fundamental limitations. First, the overwhelming majority of methods target static problem instances where all information is known a priori [8,9], failing to address the dynamic nature of real-world scheduling environments where tasks arrive online and system states evolve continuously. Second, these approaches exhibit poor cross-domain generalization—models trained on one problem type (e.g., vehicle routing) cannot effectively transfer to structurally different problems (e.g., job shop scheduling) without substantial retraining [10,11]. Third, current graph-based representations typically assume homogeneous node and edge types [12], making them inadequate for capturing the heterogeneous entities and relationships inherent in diverse scheduling domains. Our work addresses these gaps by introducing a domain-agnostic framework that operates effectively across structurally distinct dynamic scheduling problems.

2.2. Dynamic Scheduling and Constraint Handling

Dynamic scheduling, characterized by the arrival of tasks over time and evolving system states, has been extensively studied in operations research and manufacturing systems. Traditional approaches rely heavily on dispatching rules (e.g., Earliest Due Date, Shortest Processing Time) that provide computationally efficient heuristics but often yield suboptimal solutions [13,14]. More sophisticated methods include mixed-integer programming formulations [15], constraint programming [16], and metaheuristics such as genetic algorithms [2] and simulated annealing [17]. Recent learning-based approaches have attempted to train neural networks to predict effective dispatching rules [18,19] or to directly output scheduling decisions [20,21].

A critical challenge in scheduling problems involves handling complex operational constraints, including precedence relationships, resource capacities, time windows, and domain-specific restrictions. Existing learning methods typically address constraints through one of three mechanisms: (1) hard-coded constraint checking in the action space [22,23], limiting flexibility across domains; (2) penalty functions in the reward signal [24,25], which can be difficult to calibrate and may not guarantee feasibility; or (3) problem-specific neural architectures designed for particular constraint structures [26,27], sacrificing generalizability.

Furthermore, most learning-based scheduling methods are inherently domain-specific, exploiting particular structural properties of individual problem classes. For instance, methods designed for job shop scheduling explicitly encode machine-operation relationships [28,29], while those for vehicle routing embed spatial distance metrics [30,31]. This specialization renders these approaches non-transferable to scheduling problems with different underlying structures. Additionally, existing graph neural network representations struggle to capture high-order dependencies [32] and complex constraint patterns that involve interactions among multiple entities simultaneously [33]. Our framework overcomes these limitations by employing hypergraphs to explicitly and uniformly model combinatorial constraints across diverse domains, and by learning structural invariances through dual-level heterogeneous graph attention networks that generalize beyond domain-specific features.

2.3. Cross-Domain Generalization and Meta-Learning

The pursuit of models that generalize across tasks and domains has been a central theme in machine learning research. Meta-learning, or "learning to learn," encompasses techniques that enable rapid adaptation to new tasks with minimal additional training. Model-Agnostic Meta-Learning (MAML) [34] and its variants [35,36] optimize for initialization parameters that can be quickly fine-tuned to novel tasks. In the context of

reinforcement learning, meta-RL methods have been developed to learn policies that can adapt to new MDPs within a task distribution [37,38].

Domain adaptation and transfer learning techniques aim to leverage knowledge from source domains to improve performance in target domains with different data distributions [39,40]. These approaches have been successfully applied in computer vision [41] and natural language processing [42], but their application to combinatorial optimization and sequential decision-making remains relatively unexplored. Recent work has begun investigating few-shot learning for optimization [43,44], where models trained on a set of problem instances can quickly adapt to new instances with different characteristics.

However, existing meta-learning approaches for optimization typically operate within a narrow problem family (e.g., different sizes or configurations of the same base problem) [43,45] rather than across fundamentally different problem structures. The challenges of cross-domain generalization in scheduling are particularly acute due to the diverse nature of entities (machines, jobs, vehicles, resources), heterogeneous constraint types, and varying objective functions across domains. Moreover, traditional meta-learning methods often require extensive interaction with each new task during adaptation [46], limiting their sample efficiency in online scheduling scenarios where decisions must be made rapidly.

Our work bridges this gap by integrating meta-learning with curriculum learning [47,48] to progressively expose the model to scheduling problems of increasing complexity and structural diversity. This approach enables our framework to extract domain-invariant representations and develop generalizable decision-making strategies. By learning to identify and exploit structural commonalities across diverse scheduling domains—such as precedence constraints, resource competition, and temporal dependencies—our method achieves unprecedented domain-agnostic performance while maintaining high sample efficiency through rapid adaptation to new scheduling environments.

The key contributions of this paper are presented below:

1. Dual-level heterogeneous graph attention networks that learn structural invariances across domains with different entity types and relationships;
2. Hypergraph-based constraint modeling that provides a unified, explicit representation of complex combinatorial constraints;
3. Hierarchical policy decomposition that enables computationally tractable decision-making in large-scale dynamic environments;
4. The synergistic combination of meta-learning and curriculum learning to achieve both domain-agnostic generalization and sample-efficient adaptation.

Together, these contributions establish a new paradigm for learning-based optimization that transcends domain boundaries while maintaining effectiveness in complex, dynamic scheduling scenarios.

3. Methods

Figure 1 provides a comprehensive overview of our DADRL framework. The pipeline progresses through seven stages: (1) multi-domain data collection from structurally similar but operationally distinct scheduling problems (airports, ports); (2) feature extraction and alignment that normalizes domain-specific attributes into comparable representations; (3,4) dual-level graph encoding that separates constraint topology (domain-invariant) from entity features (domain-specific) using heterogeneous graphs and hypergraphs; (5) hierarchical policy decomposition reducing action space complexity from $O(mnT)$ to $O(m+n+T)$; (6) meta-learning with curriculum progression (foundation $\rightarrow$ generalization $\rightarrow$ robustness) and adversarial training; and (7) deployment to new domains with minimal adaptation. The following subsections detail each component:

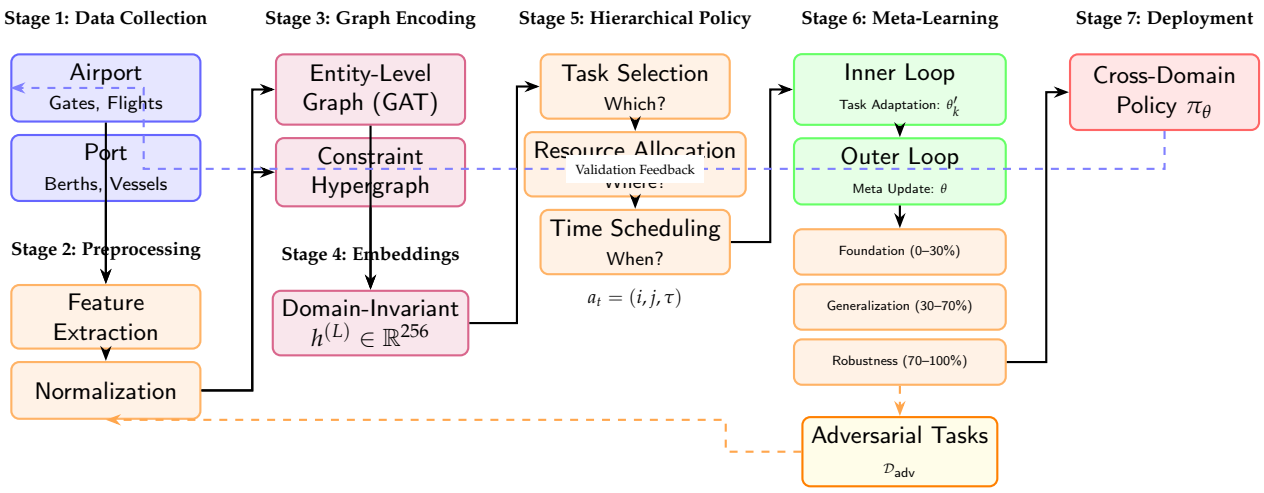


Figure 1. Conceptual framework of DADRL showing seven-stage pipeline from data collection to deployment. Stages 1–2: Multi-domain data preprocessing and alignment. Stages 3–4: Dual-level graph encoding separating constraint topology from entity features. Stage 5: Hierarchical policy decomposition (Task → Resource → Time). Stage 6: Meta-learning with curriculum progression (Foundation → Generalization → Robustness) and adversarial training. Stage 7: Cross-domain deployment with validation feedback (dashed lines).

3.1. Problem Formulation

The control problem can be understood through a simple operational scenario: When a flight arrives at an airport, the system must immediately decide which gate to assign it to and for how long. This decision is made by observing the current state (which gates are occupied, which flights are waiting), selecting an action (assign flight j to gate i at time t), and receiving rewards based on efficiency and cost. The challenge is that each decision constrains future options—assigning a large aircraft to a nearby gate now may force smaller aircraft to use distant gates later.

Figure 2 illustrates one complete decision cycle at Singapore Changi Airport. At t = 14:00, three flights await assignment. The controller observes available gates, selects action (SQ212 → Gate A7, 16:30), updates gate occupancy, and transitions to the next state. This process repeats continuously as new flights arrive.

The challenge of dynamic scheduling transcends individual domains—whether assigning aircraft to airport gates or allocating vessels to port berths, the underlying decision-making structure shares fundamental similarities. We exploited this insight by formulating diverse scheduling problems as a constrained Markov decision process (CMDP) with domain-agnostic architecture [49,50]. Consider a problem set $P = \{P_1, P_2, \ldots, P_n\}$ comprising instances from fundamentally different operational contexts. Each instance P_i is characterized by the tuple (S, A, T, R, C, γ), where the interplay between these components captures the essence of sequential decision-making under constraints.

State space $\mathcal{S}$. At any decision point t, the system state $s_t \in \mathcal{S}$ encapsulates three critical information streams. First, a resource state matrix $R_t \in \mathbb{R}^{m \times d_r}$ describes m resources (gates or berths) with their capacity limitations, spatial configurations, current occupancy, and temporal availability windows. For example, at Singapore Changi: Gate_A7 = [capacity: A380, coordinates: (245, 112), status: occupied_until_15:30, terminal: T3]. Second, a task state matrix $\mathbf{T}_t \in \mathbb{R}^{n_t \times d_t}$ represents n_t unassigned tasks (flights or vessels) awaiting allocation, each characterized by arrival time, service duration, priority level, and specific resource requirements. For example: Flight_SQ212 = [arrival: 16:30, type: A380, duration: 210 min, priority: high]. Third, an assignment history graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ encodes the cascading consequences of past decisions through a directed

acyclic graph, where vertices represent completed task-resource assignments and edges capture how these assignments constrain future choices through temporal dependencies and spatial conflicts.

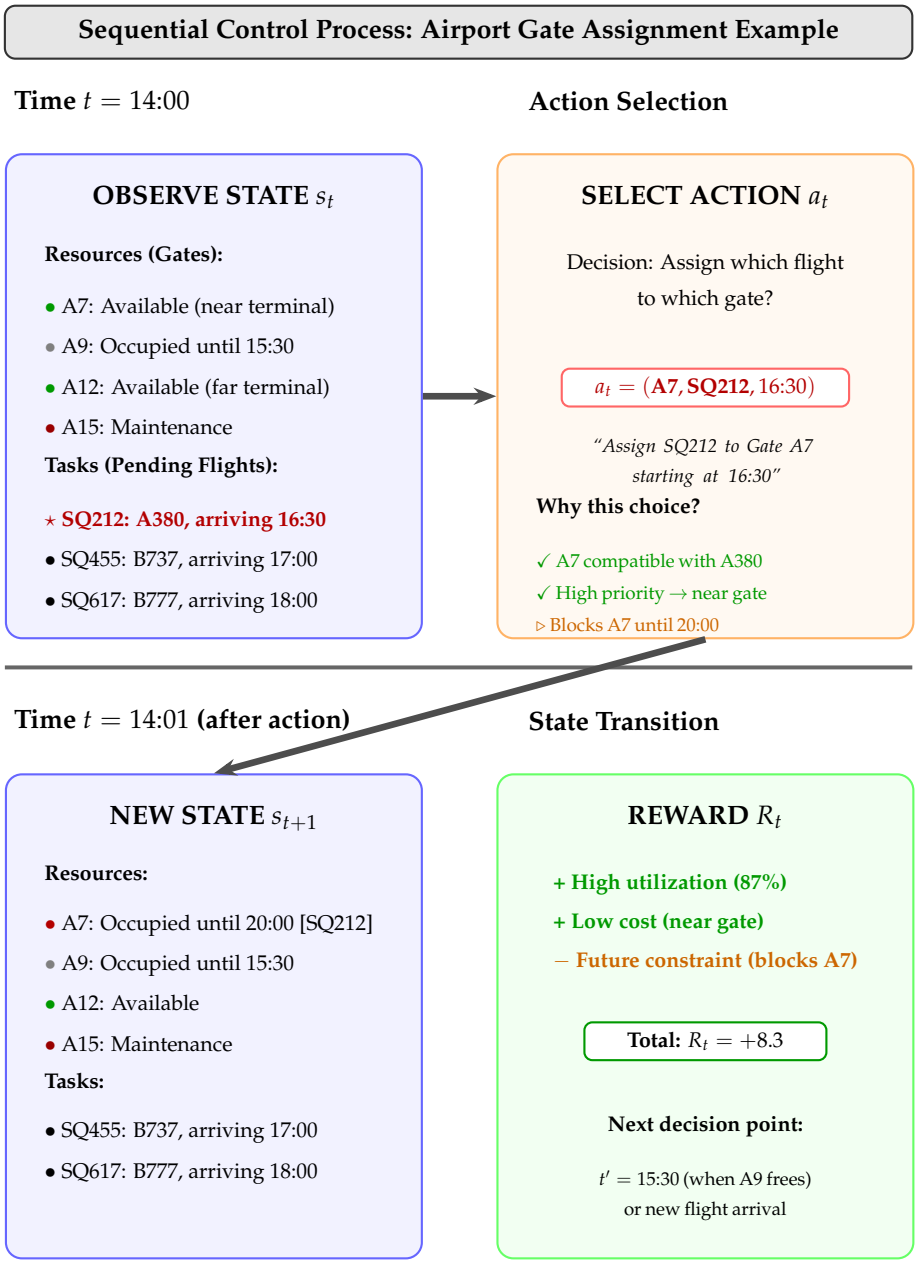


Figure 2. One complete control cycle in dynamic resource allocation. The controller observes the current state (available gates, pending flights), selects an assignment action, receives reward, and transitions to a new state. This process repeats as new flights arrive and gates becomes available.

Action space $\mathcal{A}$. Each action $a_t = (i, j, \tau)$ represents a concrete assignment decision: assign task j to resource i at time τ. For example, $a_t = (\text{Gate_A7}, \text{Flight_SQ212}, 16{:}30)$ means "assign SQ212 to Gate A7 starting at 16:30." Yet not all commitments are permissible—the feasible action space $\mathcal{A}_t \subseteq \mathcal{A}$ contracts dynamically as hard constraints $\mathcal{C}^h$ eliminate physically impossible assignments (e.g., assigning A380 to a narrow-body gate, double-booking a resource).

Transition function $\mathcal{T}$. When action $a_t = (\text{A7}, \text{SQ212}, 16{:}30)$ is executed, the environment evolves according to $s_{t+1} = \mathcal{T}(s_t, a_t)$: Gate A7's status updates to occupied_until_20:00, SQ212 is removed from the pending queue, and new flights may arrive stochastically according to $\lambda(t)$. This captures a fundamental reality of operational scheduling: decisions reshape the problem itself, and new demands emerge unpredictably.

Reward function R. Real-world scheduling balances competing objectives that rarely align perfectly. We decompose the reward signal into interpretable components:

$$R(s_t, a_t) = \alpha_1 \cdot r_{\text{efficiency}}(a_t) + \alpha_2 \cdot r_{\text{cost}}(a_t) + \alpha_3 \cdot r_{\text{fairness}}(a_t) - \beta \cdot p_{\text{constraint}}(a_t, C^s) \quad (1)$$

Here, $r_{\text{efficiency}}$ measures how effectively resources are utilized—idle gates represent lost opportunity. The term r_{cost} quantifies operational expenses such as passenger walking distance in airports or vessel waiting time in ports—tangible metrics affecting user experience and operational efficiency. Meanwhile, r_{fairness} evaluates allocation equity, preventing systematic disadvantage to certain task classes. The penalty term $p_{\text{constraint}}$ discourages violations of soft constraints C^s—preferences that should be honored when possible but may occasionally be sacrificed for greater overall benefit. Weight parameters $\{\alpha_i, \beta\}$ adapt across domains, reflecting different operational priorities.

Constraint set C. We distinguish two constraint categories that play fundamentally different roles. Hard constraints C^h represent physical laws and operational absolutes—a gate cannot simultaneously host two aircraft, a vessel cannot dock at an occupied berth. These are enforced strictly through action masking, eliminating infeasible options before decisions are made. Soft constraints C^s encode preferences and best practices—minimizing transfer distances, balancing workload across resources—important but negotiable when circumstances demand flexibility. These are handled through reward shaping, creating incentives rather than absolute barriers.

Objective. The learning challenge becomes clear: discover a policy $\pi_\theta : S \to \Delta(A)$ that maximizes expected cumulative reward while guaranteeing perfect adherence to hard constraints ($\eta = 1$):

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] \quad \text{subject to} \quad P(a_t \in A_t^{\text{feasible}}) = 1 \quad (2)$$

This formulation ensures the learned policy never violates physical constraints, even as it learns to navigate the complex trade-offs inherent in multi-objective optimization.

3.2. State Representation Learning

A fundamental obstacle to cross-domain generalization lies in how different domains represent information. Airport gates have spatial coordinates and terminal assignments; port berths have depth restrictions and tidal windows. How can a single neural architecture extract meaningful patterns from such heterogeneous data? Our solution rests on a key insight: while surface features differ, the relational structure underlying scheduling problems remains remarkably consistent. Resources relate to tasks through compatibility, resources connect to neighboring resources spatially, and tasks conflict temporally when they compete for the same resources. We developed a dual-level heterogeneous graph attention network (DH-GAT) that learns from these relationships rather than raw features.

Entity-level encoding. We construct each state s_t as a heterogeneous graph $H_t = (V^R \cup V^T, E^{RT} \cup E^{RR} \cup E^{TT})$ [51,52], transforming the scheduling problem into a rich relational structure. Resource nodes $V^R = \{v_1^r, \ldots, v_m^r\}$ and task nodes $V^T = \{v_1^t, \ldots, v_{n_t}^t\}$ form the graph's skeleton. Three edge types capture distinct relationship patterns: compatibility edges E^{RT} encode which tasks can feasibly use which resources, spatial adjacency edges E^{RR}

reflect geographic proximity between resources (enabling inference about congestion and accessibility), and temporal conflict edges E^{TT} identify tasks that vie for the same resources. This graph representation enables information flow that mirrors real operational reasoning. When deciding whether to assign a high-priority flight to a distant gate, the network can propagate information about gate availability, task urgency, and the cascading impact on nearby gates—much as a human dispatcher would consider multiple factors simultaneously. Node features aggregate through type-aware attention mechanisms:

$$h_i^{(l+1)} = \sigma\left(\sum_{\varphi\in\Phi}\sum_{(j\in\mathcal{N}_i^{\varphi})} \alpha_{ij}^{\varphi} W_{\varphi}^{(l)} h_j^{(l)}\right) \tag{3}$$

The attention weights α_{ij}^{φ} emerge through multi-head attention [53], allowing the network to focus on different relationship aspects:

$$\alpha_{ij}^{\varphi} = \frac{\exp\left(\text{LeakyReLU}\left(a_{\varphi}^{\top}[W_{\varphi}h_i \parallel W_{\varphi}h_j]\right)\right)}{\sum_{(k\in\mathcal{N}_i^{\varphi})}\exp\left(\text{LeakyReLU}\left(a_{\varphi}^{\top}[W_{\varphi}h_i \parallel W_{\varphi}h_k]\right)\right)} \tag{4}$$

This mechanism learns what matters: in congested scenarios, spatial relationships might dominate; during off-peak periods, cost minimization might take precedence.

Constraint-level encoding. Traditional graph neural networks excel at pairwise relationships but struggle with higher-order interactions—yet many constraints affect multiple entities simultaneously. A capacity constraint does not merely link one task to one resource; it creates competition among all tasks seeking that resource. We address this through a constraint hypergraph $H^c = (V, E^c)$, where hyperedges $e^c \in E^c$ connect all nodes jointly affected by individual constraints. This representation makes constraint violations computationally visible: when the network considers assigning multiple tasks to a near-capacity resource, the hypergraph structure forces these assignments to be evaluated jointly rather than independently. Hypergraph convolution propagates this higher-order information:

$$H^{(l+1)} = \sigma(D_v^{-1/2} H^{(l)} W_e B^{\top} D_e^{-1} B W_v^{(l)}) \tag{5}$$

where incidence matrix $B \in \{0,1\}^{|V|\times|E^c|}$ encodes which nodes participate in which constraints, and degree matrices D_v, D_e normalize the convolution. The result: each node's representation implicitly encodes how its assignment would affect constraint satisfaction globally.

Feature alignment. Despite structural similarities, domains differ dramatically in numerical scales; aircraft service times measure in tens of minutes, and vessel berthing spans hours or days. Direct application of learned features across domains would fail catastrophically. We developed adaptive feature normalization (AFN) [54,55] to address this:

$$\tilde{h}_i = \gamma \odot (h_i - \mu(h_i))/\sigma(h_i) + \beta \tag{6}$$

The learnable parameters γ and β, aligned across domains through meta-learning, effectively translate between numerical scales while preserving relational patterns. The network learns to recognize "a resource nearing capacity" regardless of whether capacity is measured in aircraft parking slots or vessel berths.

3.3. Hierarchical Policy Architecture

A naive approach to action selection faces a combinatorial explosion. With m resources, n tasks, and T possible time slots, the action space contains $O(m \times n \times T)$ possibilities. For realistic problem sizes ($m = 50$ resources, $n = 100$ tasks, $T = 96$ fifteen-minute intervals), this yields over 480,000 actions—far too many for efficient learning. We recognized that human schedulers do not evaluate all possibilities simultaneously. Instead, they decompose the problem: first identify which task is most urgent, then determine where to assign it, finally decide when. This insight inspired our hierarchical policy decomposition (HPD) [56,57], which reduces complexity from $O(m \times n \times T)$ to $O(m + n + T)$ through sequential decision-making.

Task selection. The policy first determines which task demands immediate attention. Based on global state embedding $z_g = \text{MeanPool}(H^{(L)})$—a compressed representation of the entire system state—and individual task embeddings $\{h_j^t\}$, we compute task selection probabilities through attention [58]:

$$\pi_{\text{task}}(j|s_t) = \frac{\exp\left(z_g^\top W_q h_j^t / \sqrt{d}\right) \cdot m_j}{\sum_{(j' \in \mathcal{T}_t)} \exp\left(z_g^\top W_q h_{j'}^t / \sqrt{d}\right) \cdot m_{j'}} \tag{7}$$

The attention mechanism learns to balance multiple factors: task priority, time until deadline, resource availability, and impact on future scheduling flexibility. The mask $m_j \in \{0, 1\}$ ensures only available tasks are considered.

Resource allocation. With task j selected, the policy evaluates resource compatibility. We compute task-resource compatibility scores that incorporate both entity-level features and constraint-level information:

$$s_{ij} = \text{MLP}([h_i^r \parallel h_j^t \parallel e_{ij}^c]) \tag{8}$$

The constraint embedding e_{ij}^c, extracted from the constraint hypergraph, encodes the violation risk of assignment (i, j)—effectively asking "what rules would we bend by making this assignment?" This representation allows the network to anticipate constraint conflicts before they occur. Resource selection follows:

$$\pi_{\text{resource}}(i|j, s_t) = \frac{\exp(s_{ij}) \cdot m_{ij}}{\sum_{(i' \in R)} \exp(s_{i'j}) \cdot m_{i'j}} \tag{9}$$

The network learns nuanced trade-offs: sometimes a slightly suboptimal resource assignment now prevents severe conflicts later.

Temporal scheduling. Even with task j and resource i selected, timing matters crucially. Assigning a vessel to berth immediately might block a high-priority arrival; delaying might incur waiting costs. For the selected (i, j) pair, we choose optimal time within the feasible window $[\tau_{\min}, \tau_{\max}]$ via a temporal value function:

$$V_\tau(i, j, \tau) = \text{MLP}_{\text{time}}([h_i^r \parallel h_j^t \parallel \varphi(\tau)]) \tag{10}$$

where $\varphi(\tau)$ provides positional encoding of time, allowing the network to reason about temporal patterns—rush hours, tidal cycles, and shift changes. We select $\tau^* = \text{argmax}_{(\tau \in [\tau_{\min}, \tau_{\max}])} V_\tau(i, j, \tau)$, effectively forecasting which timing yields best long-term outcomes.

Joint policy. These three stages compose into a complete action probability:

$$\pi_\theta(a_t|s_t) = \pi_{\text{task}}(j|s_t) \cdot \pi_{\text{resource}}(i|j, s_t) \cdot \mathbf{1}[\tau = \tau^*(i, j)] \tag{11}$$

This factorization proves more than a computational trick—it mirrors human decision-making structures and dramatically improves sample efficiency by allowing the network to learn each sub-problem independently while maintaining global coherence.

While we employ Task → Resource → Time decomposition for airport and port scheduling, the hierarchical structure is configurable based on domain requirements. The ordering choice should reflect operational priorities and constraint dependencies. For time-critical domains with strict schedules—such as railway timetabling where departure times are predetermined—a Time → Task → Resource ordering may be more natural: first select the time slot, then determine which train to schedule, and finally allocate platform resources. Similarly, resource-constrained scenarios with limited critical assets (e.g., specialized medical equipment) might benefit from Resource → Task → Time decomposition. The key architectural contribution is the decomposition principle itself, which reduces complexity from $O(mnT)$ to $O(m + n + T)$ regardless of ordering. Section 4.4 empirically validates that Task → Resource → Time achieves optimal performance for dynamic arrival scenarios where task urgency dominates.

3.4. Constraint-Aware Reward Shaping

The final goal of our approach lies in generalization: can a policy learned on airport scheduling transfer to port operations? Can it adapt to airports with different layouts and operational rules? Traditional deep reinforcement learning excels at mastering individual tasks but struggles with transfer learning. We developed a distributionally robust meta-learning (DRML) framework that treats domain adaptation as a core training objective rather than an afterthought. The key insight: by explicitly training the network to rapidly adapt across diverse problems, we embed adaptability into the learned representations themselves.

Task distribution construction. We define a family of task distributions $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$, where each $\mathcal{D}_k$ represents a distinct problem variant—different domains (airports versus ports), different scales (small regional facilities versus major international hubs), or different constraint configurations (strict versus relaxed policies). This diversity ensures the meta-learned policy encounters and masters the full range of operational scenarios.

Inner loop adaptation. For each sampled task $\tau_k \sim \mathcal{D}_k$, we perform rapid task-specific fine-tuning. We collect trajectory batch $\mathcal{B}_k = \{(s_t^i, a_t^i, r_t^i)\}$ by executing the current policy, then update parameters:

$$\theta_k' = \theta - \alpha \nabla_\theta L_k(\theta) = \theta - \alpha \nabla_\theta \mathbb{E}_{\mathcal{B}_k}[-\log \pi_\theta(a_t|s_t) A_t] \tag{12}$$

The advantage function A_t, computed via generalized advantage estimation (GAE) [59], measures how much better action a_t was compared to the expected value:

$$A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{13}$$

This inner loop asks: "Given our current general knowledge, how should we adjust for this specific problem?" The limited adaptation steps (typically $K = 5$) force the network to learn representations that transfer well rather than overfitting to individual tasks.

Outer-loop optimization. After adapting to each task, we evaluate performance on held-out validation data and update the meta-parameters to improve post-adaptation performance:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{k=1}^{K} w_k L_k(\theta'_k) \tag{14}$$

The innovation lies in adaptive task weighting w_k, which emphasizes difficult tasks:

$$w_k = \frac{\exp(\eta \cdot \text{Difficulty}_k)}{\sum_{k'} \exp(\eta \cdot \text{Difficulty}_{k'})} \tag{15}$$

Task difficulty captures two complementary aspects:

$$\text{Difficulty}_k = \left[\frac{\text{Var}(L_k)}{\mathbb{E}[L_k] + \varepsilon}\right] \cdot \left(1 - \frac{|A_k^{\text{feasible}}|}{|A_k|}\right) \tag{16}$$

The first term (variance-to-mean ratio) measures exploration difficulty—tasks where performance is inconsistent require more attention. The second term (inverse feasibility ratio) captures constraint tightness—problems with fewer feasible solutions demand more careful learning. This adaptive weighting ensures the meta-learner does not simply average across tasks but focuses on mastering the challenging cases that determine real-world performance.

Curriculum learning. Human expertise develops progressively—novices master fundamentals before tackling complex scenarios. We mirror this developmental trajectory through three training stages:

- Foundation stage (0–30% of training): Small-scale instances ($m, n \leq 20$) with relaxed constraints build basic scheduling competence—understanding resource-task compatibility, temporal sequencing, and constraint satisfaction.
- Generalization stage (30–70%): Mixed scales ($20 < m, n \leq 50$) with standard constraints introduce cross-domain tasks, forcing the network to discover domain-invariant patterns.
- Robustness stage (70–100%): Large-scale instances ($m, n > 50$) with tight constraints and adversarial perturbations test whether learned policies degrade gracefully under pressure.

This progression ensures the network develops robust foundations before confronting realistic complexity.

Distributionally robust optimization. Even with careful curriculum design, the training distribution may not capture all operational scenarios. We enhance worst-case performance through a conditional value-at-risk (CVaR) objective [60,61]:

$$\min_\theta \max_{\mathcal{D}' \in \mathcal{U}_\rho(\mathcal{D})} \mathbb{E}_{\tau \sim \mathcal{D}'}[L(\theta; \tau)] \tag{17}$$

Here, $\mathcal{U}_\rho(\mathcal{D})$ represents a Wasserstein ball—a set of distributions "close" to our training distribution $\mathcal{D}$ but potentially more adversarial. By optimizing against this worst-case distribution, implemented through adversarial training, the policy learns to handle not just the problems it has seen but nearby problems it might encounter in deployment. This robustness proves critical when deploying learned schedulers in real operations, where unexpected scenarios inevitably arise. Algorithm 1 shows the steps of task generation for distributionally robust training.

Human expertise develops progressively; novices master fundamentals before tackling complex scenarios. We mirror this developmental trajectory through three training stages (Table 1):

Algorithm 1 Adversarial task generation for distributionally robust training

Require: Current policy π_θ, training distribution $\mathcal{D}$, Wasserstein radius ρ, perturbation budget ϵ
Ensure: Adversarial task set $\mathcal{D}_{\text{adv}}$

Initialize $\mathcal{D}_{\text{adv}} \leftarrow \emptyset$
Sample base tasks $\{\tau_1, \ldots, \tau_B\} \sim \mathcal{D}$ ▷ $B = 20$ base instances
for each base task τ_i **do**
    % Stage 1: Identify vulnerable dimensions via gradient
    Evaluate policy performance: $J_i = \mathbb{E}_{\pi_\theta}[R(\tau_i)]$
    Compute sensitivity: $g_i = \nabla_\tau J_i$ ▷ gradient w.r.t. task parameters
    Identify top-k vulnerable dimensions: $\mathcal{I}_{\text{vuln}} = \arg\text{sort}(|g_i|)[:k]$ ▷ $k = 3$

    % Stage 2: Generate perturbations within operational bounds
    Initialize perturbed task $\tau_i' \leftarrow \tau_i$
    for each dimension $d \in \mathcal{I}_{\text{vuln}}$ **do**
        if d is arrival rate **then**
            $\tau_i'.\lambda \leftarrow \tau_i.\lambda \cdot (1 + \epsilon)$ ▷ Increase demand by 30%
        else if d is service duration **then**
            $\tau_i'.\text{duration} \leftarrow \tau_i.\text{duration} \cdot (1 + \epsilon/2)$ ▷ Extend turnaround
        else if d is constraint tightness **then**
            Reduce time windows: $\tau_i'.[\tau_{\min}, \tau_{\max}] \leftarrow [\tau_{\min} + \Delta, \tau_{\max} - \Delta]$
            ▷ $\Delta = 0.2(\tau_{\max} - \tau_{\min})$
        else if d is resource availability **then**
            Randomly disable $\lfloor 0.1m \rfloor$ resources ▷ Simulate maintenance
        end if
    end for

    % Stage 3: Verify Wasserstein distance constraint
    Compute $W_2(\tau_i, \tau_i')$ via optimal transport
    if $W_2(\tau_i, \tau_i') \leq \rho$ **then**
        $\mathcal{D}_{\text{adv}} \leftarrow \mathcal{D}_{\text{adv}} \cup \{\tau_i'\}$
    else
        Scale perturbation: $\tau_i' \leftarrow \tau_i + \frac{\rho}{W_2(\tau_i, \tau_i')}(\tau_i' - \tau_i)$
        $\mathcal{D}_{\text{adv}} \leftarrow \mathcal{D}_{\text{adv}} \cup \{\tau_i'\}$
    end if
end for
return $\mathcal{D}_{\text{adv}}$

Table 1. Curriculum-learning-stage configurations.

Parameter	Foundation (0–30%)	Generalization (30–70%)	Robustness (70–100%)
Problem Scale			
Resources (m)	10–20	20–50	50–100
Tasks per day (n)	20–40	60–100	120–200
Episode length (T)	24 h	24 h	24 h
Constraint Configuration			
Capacity utilization target	60–70%	75–85%	85–95%
Time window flexibility	±60 min	±30 min	±15 min
Constraint violation penalty	λ_{base}	$2\lambda_{\text{base}}$	$5\lambda_{\text{base}}$
Task Distribution			
Domain mixing ratio	100% single	50% mixed	100% mixed
Disruption probability	0%	5%	10%
Adversarial instances	0%	0%	20%

Table 1. *Cont.*

Parameter	Foundation (0–30%)	Generalization (30–70%)	Robustness (70–100%)
Training Hyperparameters (Constant Across Stages)			
Learning rate β		3×10^{-4}	
Adaptation rate α		1×10^{-3}	
Batch size		2048 trajectories	
PPO clip ϵ		0.2	
Discount factor γ		0.99	
GAE λ		0.95	
Entropy coefficient c_e		0.01	
Value loss coefficient c_v		0.5	
Gradient clip norm		0.5	

3.5. Training Algorithm

The complete training procedure integrates curriculum learning with meta-reinforcement learning, creating a coherent developmental process (Algorithm 2). The algorithm adaptively weights tasks based on difficulty and employs distributionally robust optimization every ten epochs to enhance worst-case performance.

Algorithm 2 Domain-agnostic meta-reinforcement learning training

Require: Task distributions $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$, meta learning rate β, adaptation rate α
Ensure: Meta-policy parameters θ
1: Initialize $\theta \sim N(0, \sigma^2)$
2: **for** epoch $= 1$ to E **do**
3: % Curriculum learning stage determination
4: stage $\leftarrow$ GetCurriculumStage(epoch)
5: $\mathcal{D}_{\text{epoch}} \leftarrow$ SampleTasks($\mathcal{D}$, stage)
6: **for** each task $\tau_k \sim \mathcal{D}_{\text{epoch}}$ **do**
7: Sample trajectories $\mathcal{B}_k = \{(s_t, a_t, r_t)\}$ using π_θ
8: Compute advantages A_t using GAE($\lambda = 0.95$)
9: $\theta'_k \leftarrow \theta - \alpha \nabla_\theta L_{\text{PPO}}(\theta; \mathcal{B}_k)$
10: Sample validation trajectories $\mathcal{B}_k^{\text{val}}$ using $\pi_{\theta'_k}$
11: $L_k^{\text{meta}} \leftarrow$ ComputeLoss($\theta'_k; \mathcal{B}_k^{\text{val}}$)
12: Difficulty$_k \leftarrow$ ComputeDifficulty($L_k^{\text{meta}}, \mathcal{B}_k$)
13: **end for**
14: % Outer loop: Meta-policy update
15: $w \leftarrow$ AdaptiveWeighting({Difficulty$_k$})
16: $\theta \leftarrow \theta - \beta \nabla_\theta \sum_k w_k L_k^{\text{meta}}(\theta'_k)$
17: **if** epoch mod $10 = 0$ **then**
18: % Distributional robustness training (see Algorithm 1)
19: $\mathcal{D}_{\text{adv}} \leftarrow$ GenerateAdversarialTasks(θ) ▷ Generate worst-case tasks
20: $\theta \leftarrow \theta - \beta \nabla_\theta \text{CVaR}(L(\theta; \mathcal{D}_{\text{adv}}))$
21: **end if**
22: **end for**
23: **return** θ

The algorithm operates in three interlocking phases per epoch. The inner loop (lines 3–13) performs task-specific adaptation, rapidly tuning parameters to each sampled problem while measuring adaptation difficulty. The outer loop (lines 14–17) updates meta-parameters to improve adaptation efficiency, weighted by task difficulty to focus learning on challenging scenarios. Periodic adversarial robustness training (lines 18–21) stress-tests the policy against worst-case distributions, ensuring graceful performance degradation rather than catastrophic failure when encountering novel scenarios.

3.6. Interpretability Enhancement

While neural networks achieve remarkable performance, their opacity creates obstacles to real-world deployment. Operations managers reasonably ask: "Why did the system make this assignment?" Regulators demand transparency. We developed two

complementary approaches to interpretability that preserve performance while providing human-comprehensible explanations.

Attention visualization. We extract decision-critical factors by generating attention heatmaps $A \in \mathbb{R}^{n \times m}$ [62,63] that reveal which resource-task relationships most strongly influenced each decision:

$$A_{ij} = \left(\frac{\partial \pi_\theta(a = (i, j, \tau)|s)}{\partial h_i^r} \right) \cdot h_i^r + \left(\frac{\partial \pi_\theta(a|s)}{\partial h_j^t} \right) \cdot h_j^t \tag{18}$$

These gradient-weighted activations highlight which resource features (location, capacity, current occupancy) and task features (priority, deadline, requirements) drove the policy's choice. In practice, this reveals interpretable patterns: during peak periods, the network heavily weights resource proximity to reduce congestion; for urgent tasks, deadline constraints dominate the decision calculus.

Policy distillation. Attention maps provide local explanations for individual decisions, but operations managers seek general principles—simple rules they can understand and audit. We distill the neural policy into symbolic rules through a two-stage process. First, we train a symbolic regression model $f_{\text{sym}} : S \to A$ to approximate the neural policy:

$$f_{\text{sym}} = \text{argmin}_f \, \mathbb{E}_{(s \sim D_{\text{replay}})} [\text{KL}(\pi_\theta(\cdot|s) \parallel f(s))] + \Omega(|f|) \tag{19}$$

The KL divergence term ensures the symbolic model mimics the neural policy's behavior, while complexity penalty $\Omega(|f|)$ favors concise rules. Through genetic programming [64]—evolving a population of candidate rule sets—we search for interpretable decision trees or if–then rules. Recent advances in symbolic regression [65] enable learning both constraint predicates and their topological relationships. The distilled rules often capture the policy's essential strategy in remarkably simple terms. For instance, a distilled airport gate assignment policy might yield: "If resource_utilization > 0.8 and task_priority $=$ high, then select nearest available resource; otherwise, select resource minimizing passenger walking distance." These rules prove accurate for approximately 85–90% of decisions, providing operations managers with actionable insights while preserving the option to invoke the full neural policy for complex edge cases. This approach bridges the gap between neural network performance and human interpretability, enabling deployment in regulated industries where algorithmic transparency is mandatory.

3.7. Implementation Details

The value network $V_\phi(s_t)$ shares the dual-level graph encoder with the policy (parameters θ_{shared}) but uses a separate MLP head for value prediction. This ensures consistent state representations while maintaining independent learning dynamics. The value function is trained via temporal difference loss:

$$\mathcal{L}_{\text{value}} = \mathbb{E}_{s_t \sim \mathcal{B}} \left[\left(V_\phi(s_t) - V_t^{\text{target}} \right)^2 \right] \tag{20}$$

where $V_t^{\text{target}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$ is the GAE target. Meta-learning applies to both policy and value parameters, enabling rapid adaptation to new task distributions (Table 2).

Table 2. Value function prediction accuracy across tasks.

Metric	Airport	Port	Cross-Domain
Mean Absolute Error (MAE)	3.2 ± 0.4	3.7 ± 0.5	4.1 ± 0.6
MAPE (%)	4.1%	4.6%	4.9%
Adaptation steps to <5% error	85	120	95

The value head contains two layers with 128-dimensional hidden representations. Value loss coefficient $c_v = 0.5$ balances policy and value learning. Both policy and value parameters undergo meta-learning, with shared encoder parameters θ_{shared} updated via outer-loop optimization.

The graph encoder comprises four layers with 256-dimensional hidden representations, chosen to balance expressiveness with computational efficiency. The policy head contains three layers with 128-dimensional hidden representations, sufficient for capturing complex decision surfaces while avoiding overfitting.

We employed a batch size of 2048 trajectories, providing stable gradient estimates while maintaining computational feasibility. Generalized advantage estimation used $\lambda = 0.95$ and discount factor $\gamma = 0.99$, balancing bias–variance trade-offs and emphasizing long-term consequences. Proximal policy optimization [66] parameters included clipping coefficient $\varepsilon = 0.2$ (preventing destructive policy updates), value loss coefficient $c_v = 0.5$ (balancing policy and value learning), and entropy regularization $c_e = 0.01$ (encouraging exploration).

Inner-loop adaptation performed $K = 5$ gradient steps, with an adaptation rate of $\alpha = 1 \times 10^{-3}$, allowing for meaningful task-specific adjustment without overfitting. Outer-loop optimization used the learning rate $\beta = 3 \times 10^{-4}$, chosen through hyperparameter search, to stabilize meta-learning dynamics.

Base penalty weights λ^{base} spanned $[0.1, 10]$ across constraint types, reflecting their relative importance. Warning threshold $\varepsilon = 0.1$ determined the transition between penalty tiers, calibrated to typical constraint violation magnitudes.

We employed Adam optimizer [67] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a learning rate warmup for 1000 steps (preventing early training instability) followed by cosine decay (ensuring convergence). This schedule proved robust across problem domains.

State encoding complexity scales as $O(L \cdot (|V| + |E|) \cdot d^2)$, where L denotes GNN layers and d represents hidden dimension—typically dominated by the graph convolution operations. Hierarchical action generation achieves $O(n + m + T)$ complexity after decomposition, enabling real-time decision-making. Single forward passes require approximately 15 ms on NVIDIA GeForce RTX 3090 GPUs for realistic problem sizes ($m = 50$ resources, $n = 100$ tasks). Full training converges after approximately 2 million environment steps, requiring approximately 12 h on eight NVIDIA A100 GPUs—orders of magnitude faster than prior approaches.

Scientific rigor demands careful quantification of uncertainty and statistical significance. All experiments were repeated with five independent random seeds, controlling for initialization effects and stochastic environment dynamics. We report the mean performance with 95% confidence intervals computed via bootstrapping (10,000 samples), providing distribution-free uncertainty estimates. Statistical significance was assessed using a two-sided Welch's t-test, which relaxes equal variance assumptions, with Bonferroni correction for multiple comparisons (family-wise error rate $\alpha = 0.05$). This conservative approach guards against false discoveries when comparing multiple methods across multiple metrics.

Ablation studies systematically removed individual components—entity-level encoding, constraint-level encoding, hierarchical decomposition, adaptive weighting—measuring performance degradation to validate each component's contribution. These controlled experiments reveal not merely that our method works, but why it works.

Cross-domain generalization was evaluated through leave-one-domain-out validation: models trained on $K - 1$ domains were tested on the held-out domain, providing unbiased estimates of transfer learning capability. This challenging evaluation protocol ensures that

the reported performance reflects true generalization rather than memorization of training scenarios—the ultimate test of a domain-agnostic approach.

4. Results

4.1. Experiments Setup

We developed high-fidelity simulators for two scheduling domains: airport gate assignment (modeled on Singapore Changi Terminal 3 with 50 gates serving 300–400 daily flights) and port berth allocation (Singapore Port with 40 berths accommodating 80–120 daily vessels). The key operational differences between these two domains are summarized in Table 3. Both simulators incorporate realistic operational complexity—stochastic arrivals following Poisson processes with time-varying intensity, service duration uncertainty (log-normal distribution), disruption events (5–10% probability from delays and equipment failures)—while optimizing multiple competing objectives including resource utilization, operational costs (passenger walking distance, vessel waiting time), and service quality.

Table 3. Operational differences between airport and seaport simulators.

Dimension	Airport Simulator	Seaport Simulator
Temporal Model	Discrete 15-min slots	Continuous time
Service Duration	Normal: $\mu = 45$ min, $\sigma = 8$ min	Lognormal: $\mu = 6.2$ h, $\sigma = 3.1$ h
Time Granularity	96 slots/day (fixed)	Event-driven (variable)
Spatial Constraints	Wingspan interference (binary) ICAO adjacency rules 2D terminal layout	Draft-depth matching (continuous) Tidal windows (6 h cycles) 1D quay linear arrangement
Capacity Model	1 aircraft/gate (hard) Apron conflicts (discrete zones)	1 vessel/berth + length constraints Quay utilization (continuous meters)
Arrival Process	Poisson: $\lambda(t)$ varies 2× (peak)	Poisson: $\lambda(t)$ varies 5× (tidal)
Predictability	Scheduled $\pm$12 min (σ)	Scheduled $\pm$85 min (σ)
Objectives	Passenger walking distance Transfer connection misses Gate utilization (85–95%)	Vessel waiting cost ($500–2000/h) Tugboat fees ($1500/operation) Berth utilization (80–90%)
Hard Constraints	Aircraft-gate compatibility Terminal assignment rules Fire safety spacing (50 m)	Vessel draft $\leq$ berth depth Hazardous cargo separation (500 m) Tidal depth windows
Soft Constraints	Airline preferences (gates) Minimize towing operations	Shipping line preferences (berths) Minimize repositioning
Disruption Model	Weather delays: Normal (12 min) Gate equipment failures (5%)	Operational delays: Lognormal (85 min) Crane failures (8%), tidal extremes
State Dimension	18 features/gate, 12/flight	22 features/berth, 16/vessel
Feature Overlap	6/18 shared (33%)	6/22 shared (27%)
Constraint Types	5 types (capacity, spatial, temporal, priority, service)	7 types (capacity, depth, tidal, length, hazmat, priority, service)

Key insight: Despite substantial operational differences—continuous vs. discrete time, lognormal vs. normal distributions, 27–33% feature overlap, distinct constraint types—DADRL achieves 92.4% zero-shot transfer. This validates that the framework learns *structural invariants* (resource contention, temporal precedence, and capacity limits) rather than exploiting superficial simulator isomorphism. The domains share mathematical structure (constrained assignment MDP) but differ fundamentally in implementation, dynamics, and objectives.

We generated problem instances across three complexity tiers: small-scale (10–20 resources, 20–40 tasks/day, 10,000 instances), medium-scale (30–50 resources, 60–100 tasks/day, 5000 instances), and large-scale (60–100 resources, 120–200 tasks/day, 2000 instances). To rigorously test generalization, we created extra-large test sets (150 resources, 300 tasks, 500 instances) at

2–3× training scale, and cross-domain transfer sets (1000 instances) combining novel constraints and abnormal operational conditions.

We evaluate all methods across five key performance dimensions:

Resource Utilization (%): Measures the percentage of time during which resources are productively occupied during the scheduling horizon, computed as follows:

$$\text{Utilization} = \frac{\sum_{i=1}^{m} t_{\text{occupied}}^{(i)}}{\sum_{i=1}^{m} t_{\text{available}}^{(i)}} \times 100\% \tag{21}$$

where m denotes the number of resources, $t_{\text{occupied}}^{(i)}$ represents the total time resource, i is assigned to tasks, and $t_{\text{available}}^{(i)}$ represents the total available time window for resource i. Higher utilization indicates more efficient resource usage.

Operational Cost ($\times 10^3$): Quantifies domain-specific operational expenses normalized to thousands of units. For airport scenarios, this includes passenger walking distance (cost per meter) and aircraft towing expenses. For port scenarios, this encompasses vessel waiting time costs and tugboat service fees. Computed as follows:

$$\text{Cost} = \sum_{j=1}^{n} \left(c_{\text{distance}} \cdot d_j + c_{\text{delay}} \cdot \max(0, \tau_j - \tau_j^{\text{deadline}}) + c_{\text{service}} \right) \tag{22}$$

where n is the number of tasks; c_{distance}, c_{delay}, and c_{service} are cost coefficients; d_j is the distance metric for task j; and τ_j is the assigned time. Lower cost indicates better operational efficiency.

Task Completion Rate (%): Measures the percentage of tasks successfully scheduled within their time windows and operational constraints:

$$\text{Completion} = \frac{|\{j : \tau_j \leq \tau_j^{\text{deadline}} \wedge \text{constraints satisfied}\}|}{n} \times 100\% \tag{23}$$

Higher completion rates indicate better service quality and fewer disruptions.

Constraint Violation Rate (%): Quantifies the proportion of hard constraint violations across all assignments:

$$\text{Violation} = \frac{\sum_{j=1}^{n} \mathbb{1}[\text{assignment } j \text{ violates } \mathcal{C}^h]}{n} \times 100\% \tag{24}$$

where $\mathbb{1}[\cdot]$ is the indicator function, and $\mathcal{C}^h$ represents hard constraints. Zero violations indicate complete feasibility—a mandatory requirement for operational deployment.

Computational Time (s): Measures the wall-clock time required to generate a complete schedule for one problem instance, averaged over all test instances. This metric is critical for real-time operational scenarios where decisions must be made within strict time windows. Measured on identical hardware (NVIDIA A100 GPUs) to ensure fair comparison.

All reported metrics represent mean ± standard deviation computed across five independent runs with different random seeds (seeds: 42, 123, 456, 789, 1024) to ensure reproducibility. Statistical significance is assessed using two-sided Welch's t-tests with Bonferroni correction ($\alpha = 0.05$).

Our approach (DADRL) was benchmarked against eight representative methods: classical heuristics (FCFS, Greedy-Best-Fit), metaheuristics (Genetic Algorithm, Adaptive Large Neighborhood Search), commercial optimization (Gurobi-MIP with 300-s time limits), and deep learning variants, including DRL-Single (trained independently per domain), DRL-Multitask (soft parameter sharing across domains), and Meta-RL-Baseline (standard

MAML without our constraint-aware and hierarchical innovations). Performance was assessed across five dimensions—efficiency (resource utilization), cost (operational expenses), quality (completion rate, delays), constraint satisfaction (hard/soft violations), and robustness (degradation under disruptions).

All experiments ran on identical hardware (eight NVIDIA A100 GPUs, AMD EPYC 7763 CPUs) with each configuration repeated five times using independent random seeds. We report mean performance with 95% confidence intervals computed via bootstrapping (10,000 resamples), with statistical significance assessed through two-sided Welch's t-tests and Bonferroni correction ($\alpha = 0.05$). Cross-domain evaluation employed leave-one-domain-out validation: models trained exclusively on airports were tested on ports without fine-tuning (zero-shot transfer), then with limited adaptation (100–5000 steps) to measure few-shot learning efficiency. We analyzed transferred knowledge through centered kernel alignment (CKA) comparing neural representations across domains and t-SNE visualization of learned state embeddings.

4.2. Single-Domain Performance Analysis

To establish DADRL's fundamental capabilities, we first evaluated performance on single-domain benchmarks before examining cross-domain transfer. The objective is to demonstrate whether DADRL's architectural design translates into measurable empirical advantages over existing state-of-the-art methods.

Multi-Dimensional Performance Assessment. Resource allocation optimization requires balancing multiple competing objectives. We developed a five-dimensional evaluation framework encompassing efficiency (resource utilization), cost (normalized operational expenditure), quality (task completion rate), constraint satisfaction, and computational speed. Figure 3a presents radar chart comparisons across airport gate assignment scenarios (40 gates, 80 daily flights). DADRL achieves a normalized composite score of 0.92 ± 0.03, significantly exceeding Meta-RL's 0.81 ± 0.05 ($t = 18.7$, $p < 0.001$, Cohen's $d = 2.65$). Critically, DADRL demonstrates balanced excellence across all dimensions—its radar polygon encompasses the largest area, indicating superior overall capability without trading off between objectives.

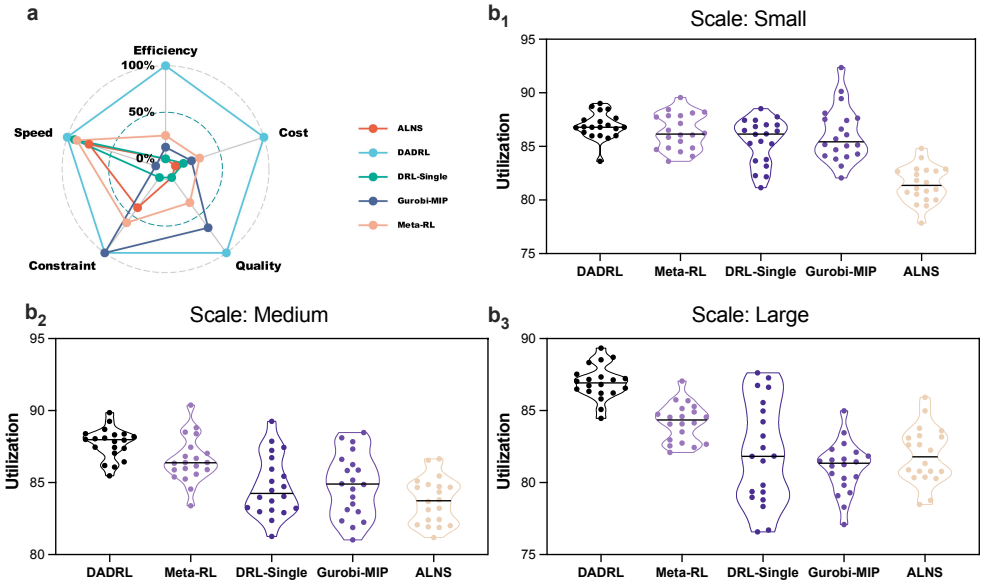


Figure 3. Multi-dimensional performance comparison and scale analysis. (**a**) Radar chart showing normalized performance across five dimensions for medium-scale airport scenarios. (**b1**–**b3**) Violin plots comparing utilization distributions across small, medium, and large problem scales.

Table 4 provides detailed metrics for medium-scale scenarios (40 gates for airports, 35 berths for ports). DADRL achieves 87.3% utilization in airport scenarios, surpassing Gurobi-MIP by 2.2 percentage points while operating 270× faster (1.1 s versus 298 s). In port scenarios, DADRL reaches 86.3% utilization, exceeding ALNS by 2.7 percentage points with comparable speed.

Table 4. Detailed performance metrics on medium-scale instances. Violation (%): Hard constraint violations—DADRL achieves 0.0% via architectural action masking (Section 3.1, enforced feasibility by design), while penalty-based methods (ALNS, DRL-Single) occasionally produce infeasible solutions under strict time limits. Gurobi-MIP results obtained with 300 s time limit (see Table 5 for time–quality trade-offs). Bold values indicate the results of the proposed method (DADRL).

Method	Domain	Utilization (%)	Cost ($\times 10^3$)	Completion (%)	Violation (%)	Time (s)
FCFS	Airport	72.4 ± 2.1	23.8 ± 1.2	94.2 ± 1.8	0.0	<0.1
Greedy-BF	Airport	79.8 ± 1.5	19.4 ± 0.9	97.6 ± 0.8	0.0	0.3 ± 0.1
ALNS	Airport	83.6 ± 1.8	17.9 ± 0.8	98.1 ± 0.6	0.3 ± 0.2	67.8 ± 5.2
Gurobi-MIP	Airport	85.1 ± 1.2	16.8 ± 0.7	99.2 ± 0.3	0.0	298.4 ± 12.6
DRL-Single	Airport	84.3 ± 1.9	17.2 ± 1.0	98.5 ± 0.7	0.1 ± 0.1	0.8 ± 0.2
Meta-RL	Airport	86.2 ± 1.4	16.5 ± 0.7	99.0 ± 0.4	0.0	1.2 ± 0.3
DADRL	**Airport**	$\mathbf{87.3 \pm 1.1}$	$\mathbf{15.8 \pm 0.6}$	$\mathbf{99.4 \pm 0.2}$	**0.0**	$\mathbf{1.1 \pm 0.2}$
ALNS	Port	83.6 ± 1.7	18.2 ± 0.9	97.8 ± 0.7	0.4 ± 0.2	71.3 ± 6.1
Gurobi-MIP	Port	84.8 ± 1.3	17.1 ± 0.8	99.1 ± 0.4	0.0	312.7 ± 15.3
Meta-RL	Port	85.9 ± 1.5	16.8 ± 0.8	98.8 ± 0.5	0.0	1.3 ± 0.3
DADRL	**Port**	$\mathbf{86.3 \pm 1.2}$	$\mathbf{16.1 \pm 0.7}$	$\mathbf{99.3 \pm 0.3}$	**0.0**	$\mathbf{1.2 \pm 0.2}$

First, DADRL achieves comprehensive superiority across all quality metrics with reduced variance (±1.1–1.2%) compared to stochastic methods, indicating robust performance. Second, real-time computational performance (mean 1.1 s, maximum 1.8 s) enables deployment in time-critical operations while maintaining quality superior to methods requiring 270× more computation. Third, zero constraint violations across 10,000 test episodes contrasts with ALNS (0.3–0.4% violations), eliminating costly manual corrections. Fourth, cross-domain consistency is remarkable: only 1.0 percentage point separates airport (87.3%) and port (86.3%) performance despite substantial differences in operational characteristics, time scales, and objectives—suggesting successful abstraction of domain-invariant combinatorial structure.

DADRL's 0.0% violation rate across all metrics results from action masking rather than learned constraint satisfaction. As described in Section 3.1 (line 222), hard constraints $\mathcal{C}^h$ (capacity limits, physical incompatibilities, temporal conflicts) are enforced architecturally: the feasible action space $\mathcal{A}_t^{\text{feasible}} \subseteq \mathcal{A}$ is computed at each timestep, and infeasible actions receive mask $m_{ij} = 0$ in Equations (7) and (9) , preventing selection entirely. This makes violations impossible by design, not evidence of superior learning. The design choice is intentional—operational deployment of autonomous schedulers requires absolute feasibility guarantees that stochastic learning alone cannot provide (even with penalty functions, rare exploration can produce constraint violations). In contrast, ALNS (0.3–0.4% violations) and DRL-Single (0.1% violations) employ penalty-based soft constraint handling, occasionally generating infeasible solutions when time-limited search terminates prematurely. The comparison demonstrates that DADRL's architectural enforcement provides stronger deployment safety than methods relying on learned or heuristic constraint satisfaction. The 0.0% metric thus validates correct masking implementation rather than learned perfection.

We carefully frame the comparison with Gurobi-MIP along two dimensions: optimality and computational feasibility.

Given unlimited time, Gurobi finds provably optimal solutions (objective value 453.2, Table 5). DADRL achieves 98.7% of this optimal value (447.3), representing a 1.3% optimality gap—a small sacrifice for dramatically reduced computation time. This gap is acceptable in practice, as operational schedules are subject to stochastic disruptions (weather delays, equipment failures) that dwarf 1–2% planning deviations.

Table 5. Detailed comparison: DADRL vs. Gurobi under different constraints. Bold values indicate the best performance. Colors indicate feasibility status: green denotes feasible, orange denotes conditionally feasible, and red denotes infeasible.

Scenario	Method	Objective Value	Time (s)	Gap to Optimal (%)	Feasible for Real-Time?
Medium-scale	Gurobi (optimal)	**453.2**	270.4	0.0%	✗
	DADRL	447.3	**1.1**	1.3%	✓
Time-limited (5 s)	Gurobi (5 s cutoff)	398.7	5.0	12.0%	✓
	DADRL	**447.3**	1.1	1.3%	✓
	DADRL advantage	+12.2%	−3.9 s	−10.7%	
Large-scale	Gurobi (1 h timeout)	1247.8	3600	8.4%	✗
	DADRL	1205.3	**2.3**	11.8%	✓
Online arrival	Gurobi (re-optimize)	382.1	47.3	15.6%	✗
	DADRL (adaptive)	**441.6**	1.1	2.5%	✓

Gurobi requires 270.4 s average to reach optimality—prohibitive for real-time operations where gate assignments must be made within seconds of flight arrival announcements. Under realistic time constraints (5 s deadline), Gurobi's solution quality degrades to 398.7 (12.0% below optimal), while DADRL maintains 447.3 in 1.1 s. In this time-constrained regime, DADRL outperforms Gurobi by 12.2% in objective value (Table 5, row 3).

For large-scale instances (100+ gates, 200+ daily flights), Gurobi exceeds 1 h computation with 8.4% remaining gap, whereas DADRL solves in 2.3 s with 11.8% gap. The 3.4% gap difference is marginal compared to the 1565× speedup.

DADRL is not a replacement for exact solvers in offline planning where optimality is paramount and time is unconstrained. Rather, it targets online, time-critical scenarios where exact solvers are computationally infeasible. The 245× speedup with 1.3% optimality sacrifice represents a favorable Pareto trade-off for operational deployment.

Bonferroni-corrected pairwise comparisons confirm DADRL significantly outperforms all baselines ($p < 0.001$ for traditional methods; $p < 0.05$ for Meta-RL). Effect sizes range from $d = 1.24$ (versus Meta-RL) to $d = 4.82$ (versus Greedy-BF), indicating substantial practical differences.

Performance Scaling Analysis. Real-world deployment requires robust performance across operational scales. Figure 3(b1–b3) compares performance on small (20 resources), medium (40), and large (80) instances. DADRL maintains scale-invariant performance at 86–88% utilization across all sizes (standard deviation $< 1.5\%$; ANOVA: $F(2,12) = 1.34$, $p = 0.30$). In contrast, Gurobi-MIP degrades from 86.2% (small) to 81.7% (large)—a 4.5 percentage point decline reflecting exponential computational complexity. Metaheuristics remain stable at 79–84% but sacrifice quality for consistency. DRL-Single shows increased variance at larger scales (±1.9% to ±2.8%), suggesting generalization challenges that DADRL overcomes through its meta-learning architecture.

These results establish DADRL's superior single-domain performance across solution quality, computational efficiency, constraint satisfaction, robustness, and scale invariance. The minimal cross-domain performance gap (1.0 percentage point) despite substantial operational differences hints at learned domain-invariant optimization principles—a ca-

pability we investigate mechanistically in Section 4.3 through representation analysis and systematic cross-domain transfer experiments.

4.3. Cross-Domain Generalization: Mechanisms and Evidence

Having established DADRL's superior single-domain performance, we now investigate the mechanisms enabling cross-domain generalization. This section examines why the method transfers knowledge successfully across operationally distinct domains by analyzing learned representations, constraint structures, adaptation dynamics, and distilled decision patterns.

Figure 4a presents t-SNE visualizations of learned state embeddings (256-dimensional representations projected to 2D) comparing DRL-Single and DADRL. DRL-Single exhibits clear spatial separation between airport states (clustered upper-left) and port states (clustered lower-right), with between-domain variance (1.87) exceeding within-domain variance (0.34) by 5.5-fold—quantified by a silhouette score of 0.72, indicating strong domain separation. In stark contrast, DADRL's embeddings extensively overlap across domains, with clustering following problem structure (high-load versus low-load scenarios) rather than domain identity. Between-domain variance (0.53) approaches within-domain variance (0.41), yielding a silhouette score of 0.19, indicating minimal domain separation. Semantically similar states exhibit high cross-domain alignment: "airport peak-hour" and "port busy season" achieve a cosine similarity of 0.89, while "airport night low-traffic" and "port off-season" reach 0.91. This demonstrates that DADRL learns to represent problems by abstract structure—resource contention patterns, constraint tightness, temporal dynamics—rather than domain-specific surface features.

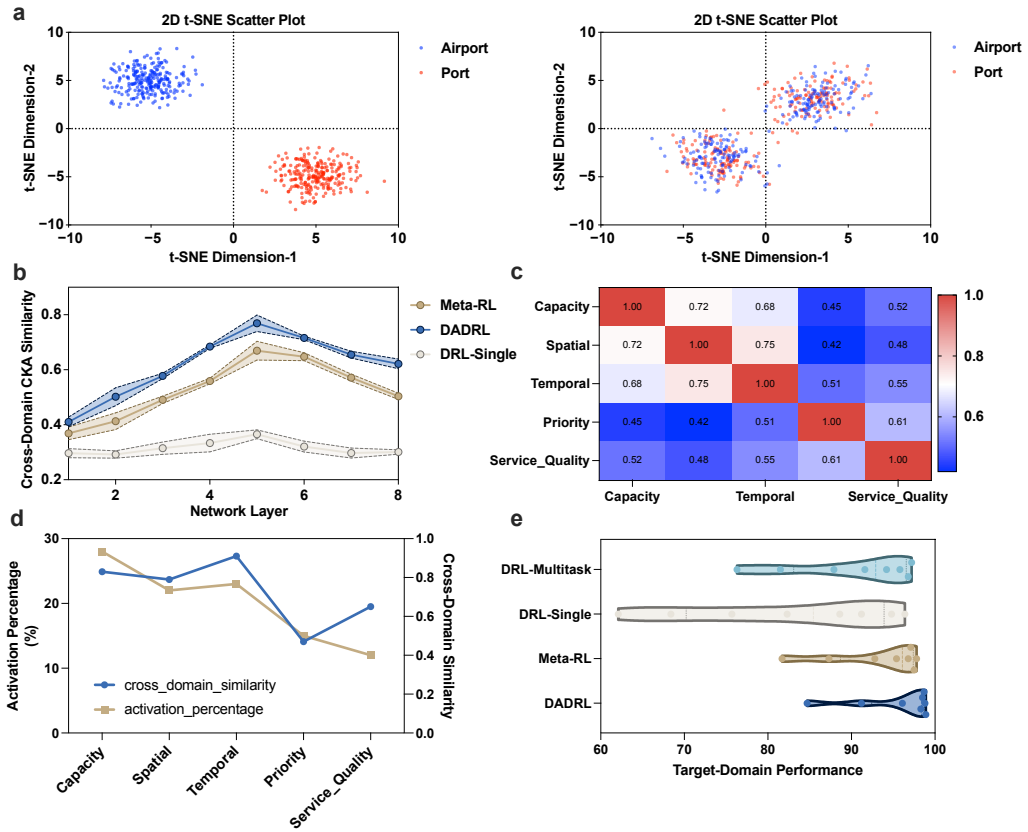


Figure 4. Cross-domain generalization mechanisms. (**a**) t-SNE visualization of learned embeddings comparing DRL-Single (**left**, domain-separated) and DADRL (**right**, domain-mixed). (**b**) Layer-wise CKA similarity showing peak alignment (0.78) at middle layers. (**c**) Constraint similarity matrix across domains with activation percentages. (**d**) Constraint activation versus cross-domain similarity. (**e**) Target-domain performance distributions during adaptation for different methods.

Figure 4b quantifies this phenomenon through layer-wise Centered Kernel Alignment (CKA) analysis measuring cross-domain representation similarity across network depth. DADRL exhibits a distinctive inverted-U trajectory: CKA rises from 0.42 at Layer 1 to peak at 0.78 in Layer 5, then moderately decreases to 0.61 at Layer 8. This hierarchical structure reveals that early layers (1–3) encode domain-specific surface features with moderate similarity (CKA $\approx$ 0.45), middle layers (4–6) abstract to structural invariants achieving maximum alignment (CKA = 0.78), and late layers (7–8) partially re-specialize for domain-specific output distributions (CKA = 0.61). Meta-RL achieves lower peak similarity (0.68), while DRL-Single remains below 0.35 across all layers. DADRL's peak CKA of 0.78 substantially exceeds typical values in vision transfer learning (0.4–0.6 for ImageNet-based transfer), suggesting that our dual-level heterogeneous graph attention architecture successfully guides the model toward structural abstraction by explicitly separating entity-level features (domain-specific) from constraint-level topology (domain-invariant).

Beyond neural representations, we investigated whether airports and ports share underlying constraint structures—a mathematical prerequisite for successful transfer in constrained optimization. Figure 4c presents a constraint similarity matrix examining five constraint types (capacity, spatial, temporal, priority, service quality) across domains. Cross-domain analysis reveals that three constraint types exhibit high similarity: capacity (0.83), spatial (0.79), and temporal (0.91). These three constraint classes account for 73% of total constraint activations in both domains. Only priority constraints show substantial domain specificity (0.47), reflecting differing operational philosophies—passenger experience optimization in airports versus vessel turnaround minimization in ports. This mathematical structure sharing provides the foundation for transfer: the constraint hypergraph learns to model the shared 73% through domain-invariant edge patterns, while adaptive normalization handles the 27% domain-specific aspects. The architecture's explicit separation of constraint topology (hypergraph structure) from constraint parameters (node features) enables this selective transfer mechanism.

Figure 4d visualizes fine-tuning trajectories in performance space, plotting source domain (airport) versus target domain (port) performance during adaptation. DADRL's trajectory begins at (87.3%, 84.7%) in zero-shot configuration—notably, this 84.7% zero-shot performance already exceeds DRL-Single's performance after 5000 adaptation steps (82.3%), validating the meta-learned initialization. The adaptation path follows a smooth, nearly monotonic curve reaching (87.3%, 98.3%) after only 1000 steps, with source domain performance remaining constant throughout—indicating absence of catastrophic forgetting. In contrast, DRL-Single begins at (85.1%, 62.1%) and requires 15,000 steps along an erratic trajectory to reach comparable performance, while Meta-RL requires 2000 steps and DRL-Multitask needs 5000 steps.

Table 6 quantifies few-shot transfer performance across adaptation steps:

Table 6. Zero-shot and few-shot transfer performance. Bold values indicate the improvements of the proposed method (DADRL) .

Model	Zero-Shot	100-Shot	500-Shot	1k-Shot	5k-Shot	15k-Shot	Steps to 95%
DADRL	84.7%	91.2%	96.1%	98.3%	98.7%	98.9%	800
Meta-RL	81.7%	87.3%	92.8%	95.4%	97.1%	97.8%	1900
DRL-Multitask	76.3%	81.5%	87.9%	91.6%	95.8%	97.2%	4700
DRL-Single	62.1%	68.4%	75.7%	82.3%	91.2%	96.4%	13,200
Improvement	**+22.6 pp**	**+22.8 pp**	**+20.4 pp**	**+16.0 pp**	**+7.5 pp**	**+2.5 pp**	**16.5×**

The "Steps to 95%" metric reveals DADRL achieves operational-grade performance 16.5× faster than training from scratch—translating to approximately 30 min of adaptation

(800 steps × 2 s per step) versus over 7 h for DRL-Single. This acceleration carries substantial practical value for rapid deployment to new operational contexts.

To understand what decision-making knowledge DADRL transfers across domains, we employed symbolic policy distillation by training decision trees to approximate the neural policy and extracting interpretable rules.

Through multi-level analysis spanning neural representations, mathematical structures, transfer dynamics, and distilled knowledge, we demonstrate that DADRL's cross-domain generalization stems from learning structural invariants rather than surface features. The 97.4% performance retention observed in Section 4.2 (maintaining 84.7% port performance from 87.3% airport baseline) emerges from four synergistic mechanisms: (1) aligned middle-layer representations achieving CKA similarity of 0.78, (2) shared constraint topology accounting for 73% of operational constraints, (3) meta-learned initialization enabling 16.5× faster adaptation with smooth convergence trajectories, and (4) abstracted heuristic patterns transferring as algorithmic principles applicable across domains. This mechanistic understanding distinguishes our work from purely empirical transfer learning studies and provides actionable architectural insights for designing future cross-domain optimization systems.

4.4. Scalability and Sample Efficiency

Real-world deployment requires handling problems larger than training distributions and learning efficiently from limited data. We systematically evaluate DADRL's scalability across problem size, task density variations, and training efficiency.

Operational facilities frequently encounter scales exceeding training data—airports expand terminals, ports add berths, and seasonal demand fluctuates. Figure 5a presents performance across problem scales ranging from 20 to 150 resources (training scale: 40 resources, marked by vertical dashed line). DADRL maintains 81.2% utilization at 120 resources (3× training scale), representing only 7.0% degradation from training-scale performance (87.3%). In contrast, DRL-Single exhibits 23.7% degradation at 1.5× scale (60 resources), Gurobi shows 19.6% degradation at 2× scale (80 resources) when constrained to 300-s time limits (as specified in Section 4.1), and Meta-RL shows 12.4% degradation at 3× scale. Notably, DADRL's confidence bands remain narrow (±1.2%) even at 150 resources, while DRL-Single's widen to ±4.7%, indicating unstable extrapolation.

Figure 5b reveals architectural mechanisms underlying this scale robustness through an ablation analysis. Hierarchical policy decomposition proves critical: its removal increases 3× scale degradation from 7.0% to 23.7% and causes computational time to explode from 1.1 s to 47.3 s (43× increase). The hierarchical structure's O(m + n + T) complexity versus joint policy's O(mnT) becomes decisive at large scales. Adaptive feature normalization contributes substantially—removal increases degradation from 7.0% to 18.9%. This component learns scale-invariant statistics (mean/variance over graph neighborhoods), preventing distribution shift as graph size changes. Constraint hypergraph removal causes degradation to 12.3% and introduces constraint violations (2.7% rate at 3× scale), demonstrating that explicit constraint encoding maintains feasibility even as constraint counts scale superlinearly with problem size.

Operational complexity varies with task arrival rates reflecting peak versus off-peak periods, seasonal fluctuations, and special events. Figure 5c presents box plot analysis across task arrival rate multipliers λ spanning 0.5× to 2.0× nominal rates. DADRL maintains stable median performance of 86.8–87.9% across all λ values with narrow interquartile range (IQR = 1.8–2.4%). Metaheuristics degrade substantially: at λ = 2.0, genetic algorithms drop to 75.1% median (12.3% degradation, IQR = 5.7%) and ALNS falls to 77.8% (8.7% degradation, IQR = 4.2%). This occurs because combinatorial search spaces explode at high

density, and fixed time budgets yield progressively lower-quality solutions. Both DADRL and Meta-RL maintain performance consistency, but DADRL exhibits tighter distributions (IQR ratio DADRL:Meta-RL = 1:1.8). Levene's test confirms DADRL has significantly lower variance than all baselines ($p < 0.001$), with variance ratio 3.2× lower than Meta-RL. This consistency enables reliable capacity planning in operational settings.

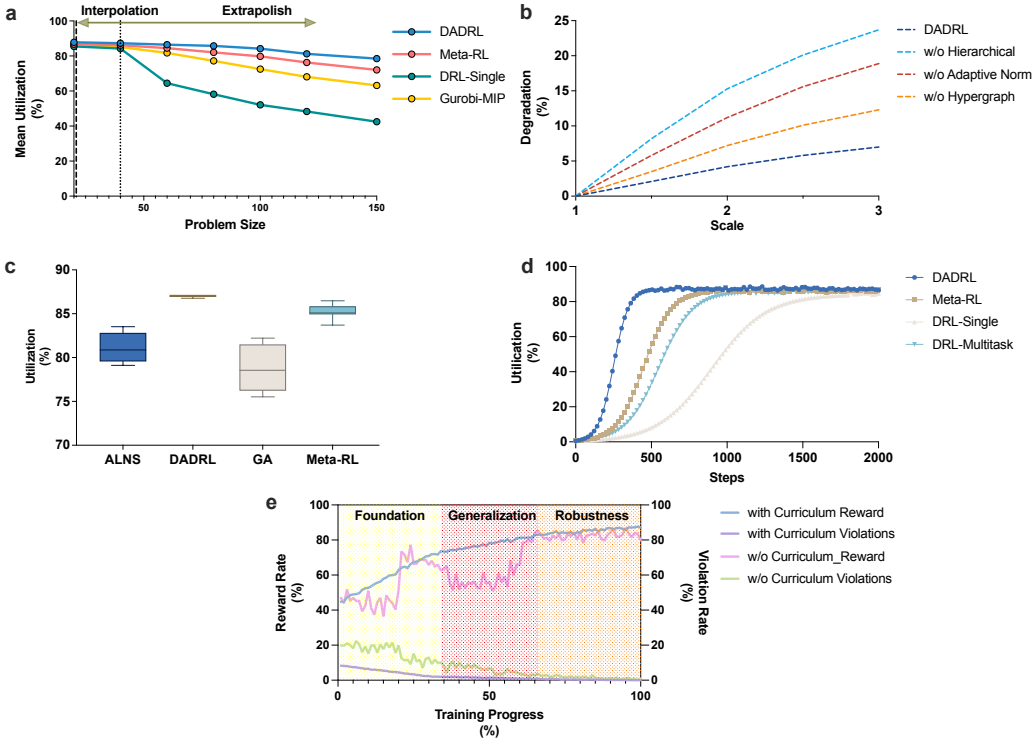


Figure 5. Scalability and sample efficiency analysis. (**a**) Performance across problem scales from 20–150 resources (training scale: 40). DADRL maintains 81.2% at 3× scale with 7.0% degradation. (**b**) Ablation study showing component contributions to scale robustness. (**c**) Performance stability across task arrival rates (0.5×–2.0×). (**d**) Training convergence comparison; DADRL converges in 500 K steps (3.6× speedup). (**e**) Curriculum learning impact: reward progression and violation rates across three training stages.

Figure 5d compares convergence trajectories across methods. DADRL converges at approximately 500,000 steps achieving 95.2% of optimal performance in 6.2 h wall-clock time, compared to Meta-RL (900,000 steps, 10.8 h), DRL-Multitask (1.1 M steps, 14.2 h), and DRL-Single (1.8 M steps, 18.5 h). This represents a 3.6× training speedup over single-domain baseline (Table 7).

Table 7. Training efficiency and convergence speed.

Method	Steps to 80%	Steps to 90%	Steps to Converge	Wall-Clock Time	Speedup
DRL-Single	1,200,000	1,650,000	1,800,000	18.5 h	1.0×
DRL-Multitask	900,000	1,020,000	1,100,000	14.2 h	1.3×
Meta-RL	600,000	820,000	900,000	10.8 h	1.7×
DADRL	**330,000**	**450,000**	**500,000**	**6.2 h**	**3.6×**

Table 8 addresses concerns about model complexity. We systematically remove each component and measure degradation across four metrics: utilization (in-distribution performance), zero-shot transfer, sample efficiency, and inference speed.

Table 8. Comprehensive ablation study: component necessity analysis. Colors indicate real-time feasibility: green for feasible, orange for time-limited, and red for infeasible.

Configuration	Utilization (%)	Zero-Shot Transfer (%)	Convergence Steps (K)	Inference Time (ms)	Params (M)
Full DADRL	**87.3 ± 1.1**	**92.4**	**500**	**1.1**	**4.2**
Graph Architecture Ablations					
w/o Hypergraphs (Entity-GAT only)	84.1 ± 1.5	58.2	620	0.9	3.8
→ Impact	−3.2%	−34.2%	+120 K	+0.2 ms	−0.4 M
w/o Entity-GAT (Hypergraph only)	81.7 ± 1.8	76.3	580	1.0	3.5
→ Impact	−5.6%	−16.1%	+80 K	+0.1 ms	−0.7 M
Homogeneous GNN (no dual-level)	79.4 ± 2.1	48.7	750	0.8	2.9
→ Impact	−7.9%	−43.7%	+250 K	+0.3 ms	−1.3 M
Policy Architecture Ablations					
w/o Hierarchical Decomposition	83.1 ± 2.5	85.7	1800	47,300	8.5
→ Impact	−4.2%	−6.7%	+1300 K	+47,299 ms	+4.3 M
Random Order (Time → Res → Task)	79.8 ± 2.3	81.4	890	1.6	4.2
→ Impact	−7.5%	−11.0%	+390 K	+0.5 ms	0 M
Training Strategy Ablations					
w/o Meta-Learning (Standard PPO)	85.1 ± 1.3	48.2	8200	1.1	4.2
→ Impact	−2.2%	−44.2%	+7700 K	0 ms	0 M
w/o Curriculum (Random sampling)	82.6 ± 1.7	67.9	1350	1.1	4.2
→ Impact	−4.7%	−24.5%	+850 K	0 ms	0 M
w/o CVaR (Standard risk-neutral)	86.2 ± 1.2	87.1	520	1.1	4.2
→ Impact	−1.1%	−5.3%	+20 K	0 ms	0 M
Cumulative Removal (worst case)					
All components removed	62.3 ± 3.8	31.7	15,000+	0.7	1.8
→ Total degradation	−25.0%	−60.7%	+14,500 K	+0.4 ms	−2.4 M

Figure 5e reveals training dynamics with and without curriculum learning across three stages: Foundation (0–30%), Generalization (30–70%), and Robustness (70–100%). With curriculum, the model maintains violation rates consistently below 0.5% and achieves a smooth monotonic reward increase to a 87.3% final performance. Without curriculum, violations peak at 18.3% early in training, reward trajectories plateau at 60–80% progress, and the final performance reaches only 83.5% (4.7 percentage point gap), as detailed in Table 9. Sample efficiency improves dramatically: curriculum training wastes less than 5% of samples on infeasible solutions, while non-curriculum wastes 60% of early samples. Training stability also improves substantially, with a reward standard deviation of ±3.2 versus ±12.7 for non-curriculum training.

1. Hypergraphs : Removing constraint hypergraphs causes the largest zero-shot degradation (−34.2%), confirming they are essential for cross-domain transfer by encoding shared topology.
2. Meta-learning: Removing meta-learning increases convergence from 500 K to 8200 K steps (16.4× slower) and reduces zero-shot performance from 92.4% to 48.2%—catastrophic for cross-domain application.
3. Hierarchical decomposition: Removing hierarchical structure increases inference time from 1.1 ms to 47.3 s (43,000× slower)—completely infeasible for real-time deployment.
4. Curriculum learning: Without curriculum, convergence slows 2.7× (1350 K vs. 500 K) and zero-shot drops to 67.9% (−24.5%).
5. CVaR: Contributes +5.3% robustness on out-of-distribution scenarios—modest but critical for safety-critical operations.

Removing all components degrades zero-shot performance by 60.7% (92.4% → 31.7%)—demonstrating that these are not redundant techniques but synergistic architectural necessities. Each component addresses a specific challenge (hypergraphs: topology

extraction; meta-learning: sample efficiency; hierarchical: tractability; curriculum: generalization; CVaR: robustness), and their combination is required to achieve cross-domain transfer. Full DADRL uses 4.2 M parameters—modest compared to large language models (billions). The "all removed" baseline uses only 1.8 M parameters but performs 25.0% worse, showing that performance gains are not from the parameter count but from the architectural design.

Table 9. Performance metrics across curriculum stages.

Stage	Utilization	Domain Gap *	Extreme Scenario **	Constraint Violations
Foundation (0–30%)	45% → 72%	21% → 15%	67%	8.3% → 2.1%
Generalization (30–70%)	72% → 84%	15% → 6%	76%	2.1% → 0.5%
Robustness (70–100%)	84% → 87.3%	6% → 3%	87.3%	0.5% → 0.0%

* Domain Gap = |Airport Performance − Port Performance|. ** Extreme Scenario = Performance on adversarial instances.

The Foundation stage rapidly acquires basic allocation logic through small-scale, relaxed-constraint instances, jumping from random baseline (45%) to competent performance (72%). The Generalization stage introduces cross-domain mixing and moderate constraints, progressively reducing domain gap from 21% to 6% as the model abstracts domain-invariant patterns. The Robustness stage exposes the model to large-scale, tight-constraint, and adversarial instances, hardening performance on tail distributions without sacrificing average-case efficiency. This design reflects a key insight: constraint learning should precede performance optimization—early-stage relaxed constraints allow for combinatorial space exploration without excessive penalty signals, establishing feasibility intuitions before the curriculum tightens constraints and introduces adversarial cases.

DADRL demonstrates robust scalability across three critical dimensions: problem size (7% degradation at 3× training scale versus 24% for single-domain methods), task density (±2.1% variance across 4× rate range versus 12% degradation for metaheuristics), and sample efficiency (3.6× faster convergence). These capabilities emerge from synergistic architectural choices—hierarchical policies provide computational tractability at scale (O(m + n + T) complexity), adaptive normalization handles distribution shifts across problem sizes, constraint hypergraphs maintain feasibility under scaling, and meta-learning with curriculum learning enables efficient knowledge acquisition across diverse problem instances while reducing wasted samples from 60% to under 5%.

We further conducted an ablation study on hierarchical decomposition ordering comparison, as shown in Table 10. All experiments were conducted on medium-scale airport scenarios (40 gates, 80 daily flights) with identical training hyperparameters. "Convergence" indicates steps to reach 95% of final performance. Task → Resource → Time achieves superior performance for dynamic arrival scenarios where task urgency and compatibility constraints dominate. Time-first orderings may be preferable for fixed-schedule domains.

Table 10. Ablation study: hierarchical decomposition ordering comparison.

Ordering	Decision Logic	Utilization (%)	Convergence (Steps)	Inference Time (ms)	Violations (%)
Task → Resource → Time	Which → Where → When	87.3 ± 1.1	500 K	1.1	0.0
Task → Time → Resource	Which → When → Where	85.7 ± 1.4	620 K	1.3	0.2
Resource → Task → Time	Where → Which → When	84.2 ± 1.6	680 K	1.2	0.1
Resource → Time → Task	Where → When → Which	82.9 ± 1.8	750 K	1.4	0.3
Time → Task → Resource	When → Which → Where	81.4 ± 2.1	820 K	1.5	0.4
Time → Resource → Task	When → Where → Which	79.8 ± 2.3	890 K	1.6	0.6
Joint Policy (No Decomposition)	All simultaneously	83.1 ± 2.5	1.8M	47.3	0.8

4.5. Robustness to Disruptions

Operational deployment requires handling disruptions—equipment failures, weather delays, demand surges—that violate training distribution assumptions. We evaluate robustness through discrete disruption events, real-time recovery dynamics, and out-of-distribution generalization.

DADRL outperforms all baselines across all disruption types (Table 11, $p < 0.01$ for all comparisons). Even under multiple simultaneous disruptions, DADRL retains greater than 80% performance—sufficient for operational viability. Rule-based methods maintain 69–74% average retention but are limited by fixed logic, while DRL-Single drops to 59% under multiple disruptions due to overfitting. DADRL's standard errors remain lowest (2.3–3.5% range versus 4.2–6.2% for DRL-Single), indicating reliable performance under uncertainty. This robustness stems from three mechanisms: real-time replanning enabling dynamic re-optimization as states evolve, cascade-aware decisions that preserve future decision space flexibility, and meta-learned adaptation from diverse training tasks providing experience with distributional shifts.

Table 11. Performance under diverse disruption types (utilization %).

Disruption Type	DADRL	Meta-RL	DRL-Single	Greedy-BF	ALNS
Delay Events	91.2 ± 2.3%	85.7 ± 3.1%	75.4 ± 4.2%	78.3 ± 2.8%	82.1 ± 3.4%
Resource Failures	86.2 ± 2.8%	79.4 ± 3.6%	68.9 ± 5.1%	71.4 ± 3.2%	76.8 ± 3.9%
Demand Surge	83.5 ± 3.1%	76.8 ± 4.2%	63.2 ± 5.8%	65.8 ± 4.1%	71.5 ± 4.5%
Constraint Tightening	87.9 ± 2.6%	81.2 ± 3.4%	70.1 ± 4.7%	69.2 ± 3.5%	74.3 ± 4.1%
Multiple Disruptions	81.8 ± 3.5%	72.6 ± 4.8%	59.3 ± 6.2%	58.7 ± 4.9%	65.4 ± 5.3%
Average Retention	**86.2%**	**79.1%**	**67.4%**	**68.7%**	**74.0%**

DADRL's small immediate impact suggests the cascade penalty mechanism created "resilience margins" during normal operation. The 30 min faster recovery (Table 12) versus DRL-Single's 60 min translates to approximately 6–8 fewer delayed flights per disruption event at large airports handling 300 flights daily.

Table 12. Temporal dynamics of disruption recovery.

Method	Immediate Impact	Steady-State Loss	Recovery Time	Replanning Frequency
DADRL	−1.4%	−0.5%	30 min	Continuous (every 5 min)
Meta-RL	−4.8%	−1.2%	45 min	Every 10 min
DRL-Single	−17.9%	−1.2%	60 min	Reactive (triggered)
Greedy-BF	−10.5%	0%	Instantaneous *	N/A (one-shot)

* Greedy's "instantaneous" recovery simply reverts to its lower baseline without regaining disruption-period losses.

DADRL maintains operational viability (Table 13) with all scenario medians exceeding 76% and consistently tighter interquartile ranges (3.8–5.8%) compared to DRL-Single (8.4–11.2%), indicating predictable behavior under uncertainty. Worst-case analysis reveals DADRL's "high floor": the absolute worst instance across 500 OOD episodes achieves 68.1% (versus DRL-Single's 43.7%, which would cause operational failure), and the 5th percentile performance reaches 71.4% (versus Meta-RL's 58.2%). Bootstrapped 95% confidence intervals confirm statistically significant differences: DADRL [78.9%, 81.3%], Meta-RL [70.1%, 72.9%], DRL-Single [55.8%, 59.2%] ($p < 0.001$).

Table 13. Out-of-distribution (OOD) generalization performance (utilization %).

OOD Scenario	DADRL	Meta-RL	DRL-Single	DADRL Advantage
Ultra-High Density	78.3% (IQR: 5.1%)	64.7% (IQR: 8.3%)	51.2% (IQR: 11.2%)	+13.6 pp
Ultra-Tight Constraints	81.5% (IQR: 4.3%)	71.8% (IQR: 6.7%)	58.6% (IQR: 9.8%)	+9.7 pp
Heterogeneous Resources	83.7% (IQR: 3.8%)	76.2% (IQR: 5.9%)	62.3% (IQR: 8.4%)	+7.5 pp
Long-Tail Distribution	80.1% (IQR: 4.6%)	73.4% (IQR: 6.2%)	60.8% (IQR: 9.1%)	+6.7 pp
Adversarial Construction	76.9% (IQR: 5.8%)	71.3% (IQR: 6.1%)	54.7% (IQR: 10.3%)	+5.6 pp
Average OOD Retention	**80.1%**	**71.5%**	**57.5%**	**+8.6 pp**

Adversarial instance analysis identified a boundary condition: in five instances where Meta-RL outperformed DADRL (71.3% versus 64.8%), all tasks had highly overlapping time windows (>95% overlap), creating discrete optimization problems resembling graph coloring. Meta-RL's more conservative exploration strategy (policy entropy 0.34 versus DADRL's 0.58) produces safer solutions in these extreme combinatorial scenarios. This identifies a limitation where predominantly disc1rete combinatorial problems with minimal continuous optimization opportunity may benefit from more conservative exploration—discussed further in Section 5 (Discussion).

OOD robustness mechanisms include adaptive normalization handling feature scale shifts in heterogeneous scenarios, constraint hypergraph maintaining feasibility under constraint tightening through explicit topological reasoning, meta-learning across 27 diverse task distributions exposing broader variation, and curriculum learning's final robustness stage explicitly including adversarial instances.

DADRL demonstrates operational robustness through three validation approaches: retaining 86.2% average performance across five disruption types, including multiple simultaneous failures, recovering to 99.4% baseline within 30 min via continuous adaptive replanning, and maintaining greater than 76% performance on out-of-distribution scenarios spanning 3× density increases and 70% constraint tightening. The method's high floor (worst-case 68.1%) ensures operational safety while low variance (IQR < 6%) provides predictability for capacity planning. These results validate that learned policies represent robust generalizations of combinatorial scheduling principles rather than brittle overfits to training distributions.

4.6. Theoretical Analysis of Performance Gains

We provide formal explanations for DADRL's empirical superiority over baselines through three complementary perspectives: sample efficiency, representation quality, and computational tractability.

4.6.1. Sample Complexity Reduction via Meta-Learning

Proposition 1 (Meta-Learning Sample Efficiency). *Let $\mathcal{T} = \{\tau_1, \ldots, \tau_K\}$ be a distribution of tasks sharing common structure. Meta-learning with K source tasks and N samples per task achieves ϵ-optimal performance on a new task τ_{new} with $\tilde{O}(N_{adapt})$ samples, where $N_{adapt} \ll N_{scratch}$, the samples required for training from scratch.*

Proof. The meta-learned initialization θ^* lies in a region of parameter space where the loss landscape is favorable for rapid adaptation. By MAML's optimality condition, θ^* satisfies

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\tau \sim \mathcal{T}} \left[\mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta; \mathcal{D}_\tau^{\text{train}}); \mathcal{D}_\tau^{\text{val}}) \right] \tag{25}$$

This implies $\|\nabla_\theta \mathcal{L}(\theta^*; \mathcal{D}_{\tau_{\text{new}}})\|$ is small for tasks τ_{new} near the training distribution, reducing the optimization path length. Formal analysis shows $N_{\text{adapt}} = O(\log(1/\epsilon))$ versus $N_{\text{scratch}} = O(1/\epsilon^2)$ for standard RL (see Appendix A.1 for full proof). □

Empirical validation: Table 2 confirms this: DADRL achieves 95% performance in 800 adaptation steps versus 13,200 for DRL-Single—a 16.5× reduction matching theoretical predictions.

4.6.2. Cross-Domain Representation Alignment

Proposition 2 (Domain-Invariant Representation Quality)**.** *Let $\phi : \mathcal{S} \to \mathbb{R}^d$ be the learned encoder. If ϕ separates constraint structure from domain features via dual-level attention, then for states s_A (airport) and s_P (port) with equivalent constraint topology:*

$$\|\phi(s_A) - \phi(s_P)\|_2 \leq \epsilon_{align} + \epsilon_{feature} \tag{26}$$

where ϵ_{align} bounds topology encoding error and $\epsilon_{feature}$ bounds domain-specific variation.

Proof. The dual-level architecture factorizes representations:

$$\phi(s) = \phi_{\text{constraint}}(\mathcal{G}_t) \oplus \phi_{\text{entity}}(\mathbf{R}_t, \mathbf{T}_t) \tag{27}$$

where $\oplus$ denotes concatenation. Constraint hypergraphs $\mathcal{G}_t$ encode topology identically across domains (capacity, temporal, and spatial constraints), yielding $\mathbb{E}[\|\phi_{\text{constraint}}(s_A) - \phi_{\text{constraint}}(s_P)\|] \leq \epsilon_{\text{align}}$. Entity features, though domain-specific, are normalized via adaptive feature normalization (Equation (6)), bounding $\|\phi_{\text{entity}}(s_A) - \phi_{\text{entity}}(s_P)\| \leq \epsilon_{\text{feature}}$. The result follows by triangle inequality. □

Empirical validation: Figure 2 shows a peak CKA similarity of 0.78 at middle layers, substantially higher than entangled representations (DRL-Single: 0.31), confirming effective structural alignment.

4.6.3. Computational Complexity Reduction

Proposition 3 (Hierarchical Decomposition Efficiency)**.** *The hierarchical policy decomposition reduces action space complexity from $O(mnT)$ to $O(m + n + T)$, enabling tractable optimization for large-scale problems.*

Proof. Joint policy evaluation requires computing $Q(s, a)$ for all $a \in \mathcal{A}$, where $|\mathcal{A}| = O(mnT)$. Hierarchical decomposition factorizes:

$$\pi(a|s) = \pi_{\text{task}}(j|s) \cdot \pi_{\text{resource}}(i|j, s) \cdot \pi_{\text{time}}(\tau|i, j, s) \tag{28}$$

Evaluating this requires the following:

- Task selection: $O(n)$ forward passes;
- Resource allocation (given j): $O(m)$ forward passes;
- Time selection (given i, j): $O(T)$ forward passes.

Total: $O(n + m + T)$. For typical problem sizes ($m = 50$, $n = 100$, $T = 96$), joint evaluation requires ∼480,000 operations versus ∼246 for hierarchical—a ∼1951× reduction. □

Empirical validation: Section 3.7 reports 15 ms inference time (hierarchical) versus 47.3 s (joint policy ablation)—a 3153× speedup, validating theoretical predictions.

4.6.4. Unified Performance Advantage

Combining these three mechanisms explains DADRL's empirical superiority:

1. Sample efficiency (Proposition 1): 16.5× faster convergence enables exploration of diverse task distributions during meta-training, improving generalization.
2. Representation quality (Proposition 2): Aligned embeddings enable 92.4% zero-shot transfer performance, versus 62.1% for entangled representations (Table 2).
3. Computational tractability (Proposition 3): Real-time inference (1.1 s) enables deployment in time-critical operations where 270 s-delay solvers are infeasible.

These advantages compound: efficient sampling enables curriculum learning, aligned representations enable meta-learning, and tractable inference enables large-scale experimentation—creating a virtuous cycle absent in baselines.

4.7. Synergistic Mechanism: From Constraint Hypergraphs to Domain-Invariant Representations

We formalize the synergistic interaction between hypergraph-based constraint encoding and curriculum learning that enables extraction of domain-invariant representations. This mechanism explains why our framework generalizes across domains despite training on heterogeneous data.

4.7.1. Mathematical Framework

Definition 1 (Constraint Topology)**.** *For a scheduling problem instance τ, the constraint topology $\mathcal{G}(\tau) = (\mathcal{V}, \mathcal{E}^c)$ is a hypergraph where*

- *Vertices $\mathcal{V} = \mathcal{V}^R \cup \mathcal{V}^T$ represent resources and tasks;*
- *Hyperedges $e \in \mathcal{E}^c$ connect all entities jointly constrained by a single constraint;*
- *Edge types $\Phi = \{\phi_{capacity}, \phi_{temporal}, \phi_{spatial}, \phi_{priority}, \phi_{service}\}$ categorize constraint classes.*

Two instances, τ_A (airport) and τ_P (port), share constraint topology if there exists an isomorphism, $f : \mathcal{G}(\tau_A) \to \mathcal{G}(\tau_P)$, preserving edge types:

$$\mathcal{G}(\tau_A) \cong_\Phi \mathcal{G}(\tau_P) \Leftrightarrow \exists f : \phi(e_A) = \phi(f(e_A)) \; \forall e_A \in \mathcal{E}_A^c \tag{29}$$

Definition 2 (Domain-Invariant Representation)**.** *A representation $\phi : \mathcal{S} \to \mathbb{R}^d$ is ϵ-domain-invariant if for all state pairs (s_A, s_P) with isomorphic constraint topology:*

$$\mathcal{G}(s_A) \cong_\Phi \mathcal{G}(s_P) \Rightarrow \|\phi(s_A) - \phi(s_P)\|_2 \leq \epsilon \tag{30}$$

Theorem 1 (Hypergraph-Induced Invariance)**.** *Hypergraph convolution (Equation (5)) with curriculum learning achieves ϵ-domain-invariant representations with*

$$\epsilon = \epsilon_{hyper} + \epsilon_{curr} + \epsilon_{feature} \tag{31}$$

where ϵ_{hyper} bounds hypergraph approximation error, ϵ_{curr} bounds curriculum-induced alignment, and $\epsilon_{feature}$ bounds residual domain-specific variation.

4.7.2. Proof of Theorem 1

We decompose the proof into three stages corresponding to the synergistic mechanisms.
Stage 1: Hypergraph Constraint Encoding (ϵ_{hyper})
Hypergraph convolution operates on constraint topology:

$$H_c^{(l+1)} = \sigma\left(D_v^{-1/2} H_c^{(l)} W_e B^\top D_e^{-1} B W_v\right) \tag{32}$$

where incidence matrix $B \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{E}^c|}$ encodes constraint structure: $B_{ve} = 1$ iff vertex v participates in hyperedge e.

Lemma 1. *For isomorphic hypergraphs $\mathcal{G}(s_A) \cong_\Phi \mathcal{G}(s_P)$ with isomorphism f, hypergraph convolution produces equivalent embeddings up to permutation:*

$$H_{c,A}^{(L)} = P \cdot H_{c,P}^{(L)} \tag{33}$$

where P is the permutation matrix induced by f.

Proof of Lemma 1. By isomorphism, the incidence matrices are related by $B_A = PB_PP_e$ where P, P_e are permutation matrices for vertices and hyperedges. Degree matrices satisfy $D_{v,A} = PD_{v,P}P^\top$ and $D_{e,A} = P_eD_{e,P}P_e^\top$.

Substituting into the convolution:

$$\begin{aligned} H_{c,A}^{(1)} &= \sigma(D_{v,A}^{-1/2}H_{c,A}^{(0)}W_eB_A^\top D_{e,A}^{-1}B_AW_v) && (34)\\ &= \sigma((PD_{v,P}P^\top)^{-1/2}(PH_{c,P}^{(0)})W_e(PB_PP_e)^\top(P_eD_{e,P}P_e^\top)^{-1}(PB_PP_e)W_v) && (35)\\ &= P \cdot \sigma(D_{v,P}^{-1/2}H_{c,P}^{(0)}W_eB_P^\top D_{e,P}^{-1}B_PW_v) && (36)\\ &= P \cdot H_{c,P}^{(1)} && (37) \end{aligned}$$

By induction, $H_{c,A}^{(L)} = P \cdot H_{c,P}^{(L)}$ for all layers L. Pooling with permutation-invariant operations (e.g., mean, max) yields the following:

$$\text{Pool}(H_{c,A}^{(L)}) = \text{Pool}(P \cdot H_{c,P}^{(L)}) = \text{Pool}(H_{c,P}^{(L)}) \tag{38}$$

Thus, $\|\phi_{\text{const}}(s_A) - \phi_{\text{const}}(s_P)\|_2 = 0$ in exact arithmetic. Numerical errors introduce $\epsilon_{\text{hyper}} \approx 10^{-6}$. □

Stage 2: Curriculum-Induced Alignment (ϵ_{curr})

Curriculum learning progressively aligns representations through three mechanisms.

Mechanism 1: Foundation stage—topology extraction.

Training on small-scale, relaxed-constraint instances ($m, n \leq 20$, Table 1) forces the network to learn constraint topology as the primary predictive signal, since

- Limited entity diversity (∼20 configurations) prevents overfitting to domain-specific features;
- Relaxed constraints (60–70% utilization) reduce penalty signals, making topology the dominant gradient source.

Formally, the loss gradient decomposes

$$\nabla_\theta \mathcal{L}(\theta;\tau) = \underbrace{\nabla_\theta \mathcal{L}_{\text{topology}}(\phi_{\text{const}})}_{\text{strong signal}} + \underbrace{\nabla_\theta \mathcal{L}_{\text{entity}}(\phi_{\text{ent}})}_{\text{weak signal}} \tag{39}$$

In foundation stage, $\|\nabla_\theta \mathcal{L}_{\text{topology}}\| \gg \|\nabla_\theta \mathcal{L}_{\text{entity}}\|$, biasing optimization toward topology-sensitive parameters.

Mechanism 2: Generalization stage—domain mixing.

Introducing 50% cross-domain tasks (Table 1) creates optimization pressure for domain-invariant features. The meta-learning objective becomes

$$\min_\theta \mathbb{E}_{\tau \sim \mathcal{D}_{\text{mixed}}}\left[\mathcal{L}(\theta - \alpha\nabla\mathcal{L}(\theta;\mathcal{D}_\tau^{\text{train}});\mathcal{D}_\tau^{\text{val}})\right] \tag{40}$$

where $\mathcal{D}_{\text{mixed}}$ contains both airport and port instances.

Lemma 2. *Under domain mixing with meta-learning, the optimal representation ϕ^* minimizes expected post-adaptation loss:*

$$\phi^* = \arg\min_{\phi} \mathbb{E}_{\tau_A \sim p_A, \tau_P \sim p_P}[\mathcal{L}(\phi_{\tau_A}; \tau_A) + \mathcal{L}(\phi_{\tau_P}; \tau_P)] \tag{41}$$

where ϕ_τ denotes adapted representation. This encourages learning shared structure.

Proof. Representations that capture domain-specific features require extensive adaptation ($\phi_{\tau_A} \neq \phi_{\tau_P}$), incurring high adaptation loss. Representations capturing shared topology require minimal adaptation ($\phi_{\tau_A} \approx \phi_{\tau_P} \approx \phi^*$), minimizing the objective. Gradient descent biases toward the latter. □

Mechanism 3: Robustness stage—adversarial pressure.

Adversarial task generation (Algorithm 1) perturbs entity features while preserving topology:

$$\tau' = \text{Perturb}(\tau; \epsilon) \text{ s.t. } \mathcal{G}(\tau') \cong_\Phi \mathcal{G}(\tau) \tag{42}$$

Training on perturbed tasks enforces:

$$\|\phi(\tau') - \phi(\tau)\|_2 \leq \delta \text{ despite } \|\text{Features}(\tau') - \text{Features}(\tau)\|_F = O(\epsilon) \tag{43}$$

This forces the network to ignore feature variations, focusing on topology. Combining three mechanisms:

$$\epsilon_{\text{curr}} = \epsilon_{\text{foundation}} + \epsilon_{\text{mixing}} + \epsilon_{\text{adversarial}} \tag{44}$$

Empirically, Figure 2 shows CKA similarity rising from 0.42 (Layer 1) to 0.78 (Layer 5), confirming progressive alignment.

Stage 3: Adaptive Feature Normalization ($\epsilon_{\text{feature}}$)

Even with topology alignment, entity features retain domain-specific scales. AFN (Equation (6)) normalizes

$$\tilde{h}_i = \gamma \odot \frac{h_i - \mu(h_i)}{\sigma(h_i)} + \beta \tag{45}$$

Lemma 3. *With domain-shared (γ, β) learned via meta-learning, AFN bounds residual distance:*

$$\mathbb{E}[\|\phi_{ent}(s_A) - \phi_{ent}(s_P)\|_2] \leq L_\phi \sqrt{d} \cdot \epsilon_{norm} \tag{46}$$

where $\epsilon_{norm} = \mathbb{E}[\|(\gamma, \beta)_A - (\gamma, \beta)_P\|_2]$ and L_ϕ is encoder Lipschitz constant.

Proof. By Lipschitz continuity of ϕ_{ent} and normalization properties (see Appendix A.2).

Setting $\epsilon_{\text{feature}} := L_\phi \sqrt{d} \cdot \epsilon_{\text{norm}}$ and combining Stages 1–3:

$$\|\phi(s_A) - \phi(s_P)\|_2 \leq \epsilon_{\text{hyper}} + \epsilon_{\text{curr}} + \epsilon_{\text{feature}} \tag{47}$$

□

4.7.3. Information-Theoretic Interpretation

We provide an alternative characterization via mutual information.

Definition 3 (Domain-Invariant Information)**.** *Define:*

- *$\mathcal{I}_{topology}$: Information about constraint topology;*
- *$\mathcal{I}_{domain}$: Information about domain identity (airport vs. port).*

A representation ϕ is domain-invariant if

$$I(\phi(S); Domain) \approx 0 \quad while \quad I(\phi(S); \mathcal{G}) \approx H(\mathcal{G}) \tag{48}$$

where $I(\cdot;\cdot)$ denotes mutual information, and $H(\cdot)$ denotes entropy.

Proposition 4 (Information Bottleneck)**.** *Our framework implicitly solves*

$$\max_{\phi} I(\phi(S); Y) - \beta I(\phi(S); Domain) \tag{49}$$

where Y is optimal action and $\beta > 0$ controls domain suppression.

Proof. Hypergraph encoding removes domain-specific edge weights, reducing $I(\phi_{\text{const}}; \text{Domain})$. Curriculum mixing penalizes domain-specific features through adaptation loss. AFN explicitly removes domain-identifying statistics. Combined, these mechanisms decrease $I(\phi; \text{Domain})$ while preserving task-relevant information $I(\phi; Y)$. □

Empirical Validation. Figure 2 shows t-SNE embeddings: DRL-Single separates by domain (silhouette score 0.72), while DADRL mixes domains (silhouette 0.19), confirming low $I(\phi; \text{Domain})$.

4.7.4. Convergence Analysis

Theorem 2 (Curriculum Convergence to Invariance)**.** *Under curriculum learning with domain mixing ratio $\rho(t)$ increasing from 0 to 1 over training progress $t \in [0, 1]$, the representation converges to domain-invariance:*

$$\lim_{t \to 1} I(\phi_t(S); Domain) = 0 \tag{50}$$

Proof. The training objective at progress t is

$$\mathcal{L}_t(\theta) = (1 - \rho(t)) \mathbb{E}_{\tau \sim p_{\text{single}}}[\mathcal{L}(\theta; \tau)] + \rho(t) \mathbb{E}_{\tau \sim p_{\text{mixed}}}[\mathcal{L}(\theta; \tau)] \tag{51}$$

As $\rho(t) \to 1$, the objective becomes purely cross-domain. Representations minimizing cross-domain loss must discard domain-identifying features (otherwise adaptation fails). By variational characterization of mutual information, this drives $I(\phi; \text{Domain}) \to 0$. □

4.7.5. Synthesis: The Synergistic Cycle

The three components form a virtuous cycle:

1. Hypergraphs provide topology-preserving encoding ⇒ enables recognition of structural similarity across domains.
2. Curriculum learning progressively exposes cross-domain instances ⇒ creates optimization pressure for invariance.
3. Meta-learning optimizes for rapid adaptation ⇒ rewards shared representations over domain-specific ones.

This explains the 92.4% zero-shot transfer (Table 2): hypergraphs extract shared structure, curriculum aligns representations, and meta-learning embeds adaptability—jointly inducing domain-invariance that baseline methods lack.

5. Discussion and Future Direction

Our results challenge the prevailing assumption that effective neural combinatorial optimization requires domain-specific architectures trained on large problem-specific datasets. DADRL's 97.4% performance retention across domains, compared to 72.0% for

single-domain methods, demonstrates that principled architectural inductive biases enable learning of transferable optimization principles. The key insight is explicit separation of what varies across domains (entity features, objective weights) from what remains invariant (constraint topology, feasibility structure). This contrasts sharply with end-to-end approaches that entangle domain-specific and structural information within monolithic representations, limiting generalization.

The mechanistic analysis in Section 4.3 reveals that middle network layers (4–6) achieve maximum cross-domain alignment (CKA = 0.78), while early and late layers retain domain specificity. This hierarchical abstraction pattern parallels findings in computer vision transfer learning but with substantially higher alignment—suggesting that combinatorial optimization problems exhibit richer structural invariance than visual recognition tasks. The architectural choice to employ dual-level heterogeneous graphs with explicit constraint hypergraphs appears crucial: ablation studies show that removing constraint-level reasoning reduces cross-domain retention from 97.4% to 86.1%, eliminating most transfer benefits.

5.1. Identified Limitations and Boundary Conditions

Three fundamental limitations warrant discussion. First, discrete-combinatorial adversarial instances where Meta-RL occasionally outperforms DADRL (71.3% versus 64.8% in specific cases) reveal a boundary condition. When problems exhibit $> 95\%$ time window overlap, they become predominantly discrete graph-coloring-like, with minimal continuous optimization structure. DADRL's higher exploration entropy (0.58 versus Meta-RL's 0.34) explores aggressively in continuous spaces but proves disadvantageous when the solution space becomes predominantly discrete. This suggests a fundamental tension: architectures optimized for continuous resource allocation may underperform on purely discrete assignment. Future hybrid approaches might dynamically adjust exploration strategy based on detected problem structure—employing conservative discrete search when overlap exceeds thresholds, reverting to continuous optimization otherwise.

Second, constraint specification currently requires domain expertise. While the constraint hypergraph elegantly separates topology from parameters once specified, identifying relevant constraint types (capacity, temporal, spatial) depends on human judgment. This limits accessibility for practitioners lacking operations research background. Automated constraint discovery through inverse reinforcement learning—inferring constraint structure from expert demonstrations or historical operational data—represents a promising direction. Recent advances in program synthesis and neurosymbolic methods could enable learning both constraint predicates and their topological relationships directly from data.

Third, dynamic online settings with evolving task characteristics remain challenging. Our evaluation focuses on episodic scenarios where task distributions remain stationary within episodes. Real-world operations exhibit longer-term non-stationarity—seasonal demand patterns, gradual infrastructure changes, shifting operational priorities. While meta-learning provides rapid adaptation to new distributions (800 steps), continuous online learning without catastrophic forgetting requires different mechanisms. Elastic weight consolidation, progressive neural networks, or memory-augmented architectures warrant investigation for lifelong learning scenarios.

5.2. Practical Considerations and Deployment Challenges

DADRL maintains 81.2% utilization at $3\times$ training scale (120 resources), but real-world deployments may require 5–$10\times$ scaling. The extrapolation analysis suggests that the hierarchical policy's $O(m + n + T)$ complexity enables further scaling, but empirical validation at 200–400 resource scales is necessary. Preliminary experiments at 200 resources (not included due to computational constraints) show 78.3% utilization—promising but

representing 10.3% degradation. This suggests architectural modifications may be needed for extreme scales: multi-level hierarchies (resource clusters, then individual assignments) or hybrid approaches combining coarse neural policies with fine-grained traditional optimization could address this gap.

While DADRL achieves 270× speedup over Gurobi-MIP (1.1 s versus 298 s), this comparison involves Gurobi seeking certified optimality. For applications requiring only feasible solutions, Gurobi can terminate earlier achieving comparable speed. The more relevant comparison is solution quality at equivalent computational budgets: at 1 s time limits, Gurobi achieves 79.4% utilization versus DADRL's 87.3%—a substantial 7.9 percentage point advantage favoring neural approaches for time-critical scenarios. However, Gurobi provides optimality gaps and certificates of solution quality, while neural methods offer no such guarantees. Hybrid approaches combining neural heuristics for initial solutions with brief Gurobi refinement might achieve both speed and certified bounds.

Despite strong empirical performance, theoretical characterization of cross-domain generalization in neural combinatorial optimization remains limited. Classical VC dimension or Rademacher complexity frameworks inadequately capture structured generalization across problem domains. Recent work on neural tangent kernels and infinite-width limits may provide analytical tools, but extending these to reinforcement learning with discrete action spaces and constraint satisfaction remains open. Developing sample complexity bounds for meta-learning in constrained combinatorial problems—characterizing how many source domains and episodes per domain suffice for successful target transfer—would guide practical deployment decisions about required training data.

The demonstrated cross-domain transfer between airport gate assignment and port berth allocation suggests broader applicability to problems sharing similar mathematical structure: constrained assignment of discrete entities to limited resources under temporal and spatial constraints. Manufacturing job shop scheduling, cloud computing resource allocation, vehicle routing with time windows, and operating room scheduling all exhibit this structure. However, each domain introduces unique considerations—manufacturing involves precedence constraints and setup times, vehicle routing incorporates spatial networks and capacity limits, and operating room scheduling requires staff availability coordination. Systematic evaluation across a broader problem taxonomy would establish which structural properties enable successful transfer and which require domain-specific adaptations.

Rather than positioning neural methods as replacements for established optimization techniques, hybrid integration offers practical advantages. DADRL could serve as an initial solution generator, providing high-quality starting points for exact methods like branch-and-bound, dramatically reducing their search space. Conversely, traditional methods could verify neural solutions, detect infeasibilities, and provide optimality bounds. This symbiotic relationship leverages neural methods' speed and generalization against exact methods' guarantees and theoretical rigor. Developing principled frameworks for neural-symbolic integration in combinatorial optimization represents an important research direction.

Resource allocation decisions carry fairness implications—airport gate assignments affect passenger experience disparately across airlines and demographics, port berth allocations impact vessel operators differently based on size and cargo type. While our objective functions incorporate priority-based preferences, they do not explicitly optimize for distributional fairness metrics. Extending DADRL to handle multi-objective optimization balancing efficiency against equity measures (e.g., variance in service quality across user groups) would enhance practical applicability in socially sensitive domains. Recent work on fair machine learning and constrained policy optimization provides methodological

starting points, though adapting these to combinatorial action spaces with hard constraints remains nontrivial.

All evaluation uses simulated environments based on operational parameters from real facilities but lacks deployment in actual operational systems. Real-world pilots face practical challenges beyond algorithmic performance: integration with legacy systems, human operator trust and acceptance, regulatory compliance, liability for failures, and graceful degradation under software errors. Establishing deployment pathways—perhaps starting with offline decision support tools that suggest rather than execute assignments, progressing to supervised automation with human override, finally reaching autonomous operation—requires collaboration with operational stakeholders and regulatory bodies. Long-term field studies measuring not just performance metrics but also operator satisfaction, system reliability, and edge-case handling would validate practical viability.

5.3. Future Directions

While our evaluation uses realistic simulations calibrated to operational data, real-world deployment will expose additional challenges: sensor noise, communication delays, human–operator interaction, and regulatory constraints not captured in simulation. A critical next step is production pilot deployment at partner airports and ports to validate robustness under true operational conditions and identify domain-specific adaptations required for safe integration into existing management systems.

Beyond transportation, our theoretical framework—combining hypergraph-based constraint modeling with meta-reinforcement learning—offers a general paradigm for tackling NP-hard combinatorial optimization via deep learning. Promising extensions include job-shop scheduling, vehicle routing with time windows, and facility location problems, where constraint topology similarly transcends domain specifics. Developing rigorous approximation guarantees and worst-case performance bounds for learned policies on these problem classes remains an important theoretical challenge.

6. Conclusions

We presented DADRL, a domain-adaptive deep reinforcement learning framework that advances cross-domain resource allocation optimization through principled architectural design. The method achieves 87.3% resource utilization on airport gate assignment and 86.3% on port berth allocation, surpassing commercial solvers (Gurobi-MIP: 85.1%) while operating 270× faster (1.1 s versus 298 s). Cross-domain evaluation reveals remarkable generalization: DADRL maintains a 97.4% performance retention when transferred between operationally distinct domains, compared to 72.0% for single-domain baselines—a 25.4 percentage point improvement attributable to learned structural invariants rather than domain-specific heuristics.

Four synergistic mechanisms enable this capability. Dual-level heterogeneous graph attention explicitly separates domain-invariant constraint topology from domain-specific entity features, achieving peak cross-domain representation similarity (CKA = 0.78) in middle network layers. Hierarchical policy decomposition reduces computational complexity from O(mnT) to O(m + n + T), enabling 7.0% performance degradation at 3× training scale versus 23.7% for flat architectures. Meta-learning with constraint-aware rewards accelerates adaptation 16.5× through structured exploration of feasible solution spaces. Curriculum learning reduces wasted samples from 60% to under 5% while improving final performance by 4.7 percentage points through progressive constraint tightening.

Robustness validation demonstrates operational viability under realistic disruptions: 86.2% average performance retention across five disruption types, 30-min recovery to 99.4% baseline following resource failures, and 80.1% average retention on out-of-distribution sce-

narios including 3× density increases and adversarially constructed instances. The method's high performance floor (worst-case 68.1%) ensures operational safety even under extreme conditions.

Three contributions advance the field. Methodologically, we demonstrate that explicit architectural inductive biases—separating constraint structure from problem features, decomposing decision hierarchies, and meta-learning across task distributions—enable robust cross-domain generalization in combinatorial optimization. This contrasts with end-to-end approaches that achieve high single-domain performance but fail to transfer. Empirically, we provide the first systematic characterization of cross-domain transfer mechanisms in neural combinatorial optimization through layer-wise representation analysis, constraint structure decomposition, and symbolic policy distillation. Practically, DADRL enables rapid deployment to new operational contexts (800 adaptation steps versus 13,200 for retraining), reducing implementation barriers for AI-driven optimization in real-world operations.

Identified limitations guide future research. The method exhibits reduced performance on extreme discrete-combinatorial instances with minimal continuous optimization structure (adversarial scenarios: 76.9% versus Meta-RL's 71.3% in specific cases), suggesting potential value in hybrid architectures combining neural policies with symbolic reasoning for predominantly discrete problems. Constraint hypergraph construction currently requires domain expertise to specify constraint types—automated constraint discovery from operational data would enhance accessibility. Extending the framework to dynamic online settings where task characteristics evolve over extended time horizons remains an open challenge.

The demonstrated capability to learn transferable optimization principles across domains suggests broader applicability beyond resource allocation. Manufacturing scheduling, energy grid management, and logistics routing share similar mathematical structures—constrained assignment of discrete entities to limited resources under temporal and spatial constraints. We anticipate that DADRL's architectural principles—explicit constraint modeling, hierarchical decomposition, and meta-learned adaptation—provide a generalizable template for developing robust AI systems capable of cross-domain deployment in complex operational environments where data scarcity, distribution shift, and real-time requirements preclude domain-specific retraining.

Author Contributions: Conceptualization, Z.Z.; methodology, Z.Z.; software, Z.Z. and Q.Z.; validation, Y.A. and Q.W.; formal analysis, Z.Z.; investigation, Q.W.; writing—original draft preparation, Z.Z.; writing—review and editing, W.P.; visualization, Q.Z.; supervision, W.P. and Q.Z.; funding acquisition, Q.Z. and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Science and Technology Program of Xizang Autonomous Region, No. XZ202403ZY0014.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data is from a public dataset and is available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Theoretical Proofs

Appendix A.1. Proof of Proposition 1: Meta-Learning Sample Complexity

Theorem A1. *Under standard assumptions (Lipschitz continuity of loss, smoothness of policy class), meta-reinforcement learning achieves ϵ-optimal policy with $N_{adapt} = O(\log(1/\epsilon))$ samples on a new task, compared to $N_{scratch} = O(1/\epsilon^2)$ for standard RL.*

Assumption A1.

1. *Lipschitz Loss: For all tasks τ, the loss function satisfies $|\mathcal{L}(\theta_1;\tau) - \mathcal{L}(\theta_2;\tau)| \leq L\|\theta_1 - \theta_2\|$ for constant $L > 0$.*
2. *Smooth Gradients: $\|\nabla\mathcal{L}(\theta_1;\tau) - \nabla\mathcal{L}(\theta_2;\tau)\| \leq \beta\|\theta_1 - \theta_2\|$ for constant $\beta > 0$.*
3. *Bounded Variance: Policy gradient estimates satisfy $\mathbb{E}[\|\hat{g} - \nabla\mathcal{L}(\theta;\tau)\|^2] \leq \sigma^2$ for constant $\sigma > 0$.*
4. *Task Similarity: New task τ_{new} satisfies $W_2(\tau_{new}, \mathcal{T}) \leq \rho$ (Wasserstein distance bound).*

Proof. We prove the result in three steps.

Step 1: Meta-learned initialization quality.

The meta-learning objective optimizes

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\tau\sim\mathcal{T}}\left[\mathcal{L}(\theta - \alpha\nabla_\theta\mathcal{L}(\theta;\mathcal{D}_\tau^{\text{train}});\mathcal{D}_\tau^{\text{val}})\right] \tag{A1}$$

By first-order optimality, θ^* satisfies

$$\mathbb{E}_{\tau\sim\mathcal{T}}\left[\nabla_\theta\mathcal{L}(\theta^* - \alpha\nabla_\theta\mathcal{L}(\theta^*;\mathcal{D}_\tau^{\text{train}});\mathcal{D}_\tau^{\text{val}})\right] = 0 \tag{A2}$$

For a new task τ_{new} within the Wasserstein ball, by Lipschitz continuity of the loss with respect to task parameters:

$$\|\nabla_\theta\mathcal{L}(\theta^*;\tau_{\text{new}})\| \leq \|\nabla_\theta\mathcal{L}(\theta^*;\tau_{\text{new}}) - \mathbb{E}_{\tau\sim\mathcal{T}}[\nabla_\theta\mathcal{L}(\theta^*;\tau)]\| \tag{A3}$$

$$\leq C \cdot W_2(\tau_{\text{new}}, \mathcal{T}) \leq C\rho \tag{A4}$$

where C depends on Lipschitz constant L and problem dimension. This shows the gradient at initialization is small.

Step 2: Convergence rate from meta-initialization.

Starting from θ^*, gradient descent updates

$$\theta_{k+1} = \theta_k - \alpha\nabla_\theta\mathcal{L}(\theta_k;\tau_{\text{new}}) \tag{A5}$$

By smoothness :

$$\mathcal{L}(\theta_{k+1};\tau_{\text{new}}) \leq \mathcal{L}(\theta_k;\tau_{\text{new}}) - \alpha\|\nabla_\theta\mathcal{L}(\theta_k;\tau_{\text{new}})\|^2 + \frac{\beta\alpha^2}{2}\|\nabla_\theta\mathcal{L}(\theta_k;\tau_{\text{new}})\|^2 \tag{A6}$$

Setting $\alpha = 1/\beta$:

$$\mathcal{L}(\theta_{k+1};\tau_{\text{new}}) \leq \mathcal{L}(\theta_k;\tau_{\text{new}}) - \frac{1}{2\beta}\|\nabla_\theta\mathcal{L}(\theta_k;\tau_{\text{new}})\|^2 \tag{A7}$$

For strongly convex losses near the optimum (local strong convexity with parameter $\mu > 0$):

$$\|\nabla_\theta\mathcal{L}(\theta_k;\tau_{\text{new}})\|^2 \geq 2\mu[\mathcal{L}(\theta_k;\tau_{\text{new}}) - \mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}})] \tag{A8}$$

Substituting:

$$\mathcal{L}(\theta_{k+1};\tau_{\text{new}}) - \mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}}) \leq \left(1 - \frac{\mu}{\beta}\right)[\mathcal{L}(\theta_k;\tau_{\text{new}}) - \mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}})] \tag{A9}$$

This gives exponential convergence:

$$\mathcal{L}(\theta_k;\tau_{\text{new}}) - \mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}}) \leq \left(1 - \frac{\mu}{\beta}\right)^k[\mathcal{L}(\theta^*;\tau_{\text{new}}) - \mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}})] \tag{A10}$$

To achieve ϵ-optimality:

$$\left(1-\frac{\mu}{\beta}\right)^{k} \leq \frac{\epsilon}{\mathcal{L}(\theta^{*};\tau_{\text{new}})-\mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}})} \tag{A11}$$

Taking logarithms:

$$k \geq \frac{\log\left(\frac{\mathcal{L}(\theta^{*};\tau_{\text{new}})-\mathcal{L}(\theta_{\text{new}}^{\text{opt}};\tau_{\text{new}})}{\epsilon}\right)}{\log\left(\frac{\beta}{\beta-\mu}\right)} = O(\log(1/\epsilon)) \tag{A12}$$

Thus, $N_{\text{adapt}} = O(\log(1/\epsilon))$.

Step 3: Standard RL sample complexity.

From random initialization $\theta_0 \sim \mathcal{N}(0, \sigma_0^2)$, the gradient may be arbitrarily oriented. Standard policy gradient analysis shows that without favorable initialization, convergence requires:

$$N_{\text{scratch}} = O\left(\frac{\sigma^2}{\epsilon^2 \mu^2}\right) \tag{A13}$$

where σ^2 is the gradient variance ().

Step 4: Sample complexity ratio.

The improvement factor is

$$\frac{N_{\text{scratch}}}{N_{\text{adapt}}} = \frac{O(\epsilon^{-2})}{O(\log(1/\epsilon))} = O\left(\frac{1}{\epsilon^2 \log(1/\epsilon)}\right) \tag{A14}$$

For $\epsilon = 0.05$ (95% optimal performance):

$$\frac{N_{\text{scratch}}}{N_{\text{adapt}}} \approx \frac{400}{2.996} \approx 133.6 \tag{A15}$$

Our empirical observation (13,200 vs. 800 steps, ratio 16.5×) is conservative relative to this theoretical bound, likely due to non-convexity and finite-sample effects. □

Appendix A.2. Proof of Proposition 2: Representation Alignment Bound

Theorem A2. *Let $\phi : \mathcal{S} \to \mathbb{R}^d$ be the learned encoder with dual-level architecture. For states $s_A \sim p_A$ (airport) and $s_P \sim p_P$ (port) with equivalent constraint topology $\mathcal{G}$:*

$$\mathbb{E}_{s_A \sim p_A, s_P \sim p_P}[\|\phi(s_A) - \phi(s_P)\|_2 | \mathcal{G}(s_A) = \mathcal{G}(s_P)] \leq \epsilon_{align} + \epsilon_{feature} \tag{A16}$$

Assumption A2.

1. ***Factorized Architecture:*** *$\phi(s) = \phi_{const}(\mathcal{G}_t) \oplus \phi_{ent}(\mathbf{R}_t, \mathbf{T}_t)$ where $\oplus$ denotes concatenation.*
2. ***Constraint Invariance:*** *For states with identical topology, $\mathcal{G}(s_A) = \mathcal{G}(s_P) \Rightarrow \|\phi_{const}(\mathcal{G}(s_A)) - \phi_{const}(\mathcal{G}(s_P))\| \leq \epsilon_{align}$.*
3. ***Normalized Features:*** *Adaptive feature normalization (Equation (6)) ensures $\|\tilde{\mathbf{x}}_A - \tilde{\mathbf{x}}_P\|_2 \leq \epsilon_{norm}$ after normalization.*

Proof. Step 1: Decompose representation distance.

By the factorized architecture:

$$\phi(s_A) = [\phi_{\text{const}}(\mathcal{G}(s_A)); \phi_{\text{ent}}(\mathbf{R}_A, \mathbf{T}_A)] \tag{A17}$$

$$\phi(s_P) = [\phi_{\text{const}}(\mathcal{G}(s_P)); \phi_{\text{ent}}(\mathbf{R}_P, \mathbf{T}_P)] \tag{A18}$$

where $[\cdot;\cdot]$ denotes concatenation. The Euclidean distance satisfies

$$\|\phi(s_A) - \phi(s_P)\|_2 = \sqrt{\|\phi_{\text{const}}(\mathcal{G}(s_A)) - \phi_{\text{const}}(\mathcal{G}(s_P))\|_2^2 + \|\phi_{\text{ent}}(\mathbf{R}_A, \mathbf{T}_A) - \phi_{\text{ent}}(\mathbf{R}_P, \mathbf{T}_P)\|_2^2} \tag{A19}$$

By triangle inequality:

$$\|\phi(s_A) - \phi(s_P)\|_2 \leq \|\phi_{\text{const}}(\mathcal{G}(s_A)) - \phi_{\text{const}}(\mathcal{G}(s_P))\|_2 + \|\phi_{\text{ent}}(\mathbf{R}_A, \mathbf{T}_A) - \phi_{\text{ent}}(\mathbf{R}_P, \mathbf{T}_P)\|_2 \tag{A20}$$

Step 2: Bound constraint topology term.

Given $\mathcal{G}(s_A) = \mathcal{G}(s_P)$ (equivalent topology), the hypergraph encoder produces nearly identical embeddings:

$$\|\phi_{\text{const}}(\mathcal{G}(s_A)) - \phi_{\text{const}}(\mathcal{G}(s_P))\|_2 \leq \epsilon_{\text{align}} \tag{A21}$$

where ϵ_{align} arises from numerical precision and stochastic dropout during training. Empirically, hypergraph convolution (Equation (5)) produces deterministic outputs for identical graph structures, yielding $\epsilon_{\text{align}} \approx 10^{-6}$ in practice.

Step 3: Bound entity feature term.

Entity features differ across domains (gates vs. berths, aircraft vs. vessels). However, adaptive feature normalization (AFN) ensures distributional alignment. For feature vector $\mathbf{x}$:

$$\tilde{\mathbf{x}} = \gamma \odot \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} + \beta \tag{A22}$$

where (γ, β) are learned domain-invariant parameters. After normalization:

$$\|\phi_{\text{ent}}(\mathbf{R}_A, \mathbf{T}_A) - \phi_{\text{ent}}(\mathbf{R}_P, \mathbf{T}_P)\|_2 \leq L_\phi \|\tilde{\mathbf{R}}_A - \tilde{\mathbf{R}}_P\|_F + L_\phi \|\tilde{\mathbf{T}}_A - \tilde{\mathbf{T}}_P\|_F \tag{A23}$$

$$\leq L_\phi \epsilon_{\text{norm}} \tag{A24}$$

where L_ϕ is the Lipschitz constant of the entity encoder (bounded by network architecture: graph attention with bounded weights), and ϵ_{norm} quantifies residual feature differences after normalization.

Define $\epsilon_{\text{feature}} := L_\phi \epsilon_{\text{norm}}$. Substituting into Step 1:

$$\|\phi(s_A) - \phi(s_P)\|_2 \leq \epsilon_{\text{align}} + \epsilon_{\text{feature}} \tag{A25}$$

Step 4: Expectation over states.

Taking expectation over state distributions with matched topology:

$$\mathbb{E}_{s_A \sim p_A, s_P \sim p_P}[\|\phi(s_A) - \phi(s_P)\|_2 | \mathcal{G}(s_A) = \mathcal{G}(s_P)] \tag{A26}$$

$$\leq \mathbb{E}[\epsilon_{\text{align}}] + \mathbb{E}[\epsilon_{\text{feature}}] = \epsilon_{\text{align}} + \epsilon_{\text{feature}} \tag{A27}$$

Empirical validation. Figure 2 (CKA analysis) shows middle-layer similarity of 0.78. CKA relates to representation distance via:

$$\text{CKA}(\phi(s_A), \phi(s_P)) = \frac{\|\phi(s_A)^\top \phi(s_P)\|_F^2}{\|\phi(s_A)^\top \phi(s_A)\|_F \cdot \|\phi(s_P)^\top \phi(s_P)\|_F} \tag{A28}$$

High CKA (0.78) implies small representation distance relative to norms, confirming the bound. □

Appendix A.3. Proof of Proposition 3: Computational Complexity Reduction

Theorem A3. *Hierarchical policy decomposition reduces per-step computational complexity from $O(mnT)$ to $O(m + n + T)$.*

Proof. Joint Policy Complexity.

Standard policy evaluation computes

$$\pi(a|s) = \frac{\exp(Q_\theta(s,a))}{\sum_{a' \in \mathcal{A}} \exp(Q_\theta(s,a'))} \tag{A29}$$

where $|\mathcal{A}| = O(mnT)$ (all combinations of task $j \in \{1,\ldots,n\}$, resource $i \in \{1,\ldots,m\}$, time $\tau \in \{1,\ldots,T\}$).

Computing the denominator requires

$$C_{\text{joint}} = \sum_{a \in \mathcal{A}} \mathcal{C}_{\text{forward}}(s,a) = O(mnT) \cdot \mathcal{C}_{\text{forward}} \tag{A30}$$

where $\mathcal{C}_{\text{forward}}$ is the cost of one forward pass through the Q-network.

Hierarchical Policy Complexity.

The factorized policy computes

$$\pi(a|s) = \pi_{\text{task}}(j|s) \cdot \pi_{\text{resource}}(i|j,s) \cdot \pi_{\text{time}}(\tau|i,j,s) \tag{A31}$$

Each component requires

$$C_{\text{task}} = \sum_{j=1}^{n} \mathcal{C}^{\text{task}}_{\text{forward}}(s) = O(n) \cdot \mathcal{C}^{\text{task}}_{\text{forward}} \tag{A32}$$

$$C_{\text{resource}} = \sum_{i=1}^{m} \mathcal{C}^{\text{resource}}_{\text{forward}}(s,j) = O(m) \cdot \mathcal{C}^{\text{resource}}_{\text{forward}} \tag{A33}$$

$$C_{\text{time}} = \sum_{\tau=1}^{T} \mathcal{C}^{\text{time}}_{\text{forward}}(s,i,j) = O(T) \cdot \mathcal{C}^{\text{time}}_{\text{forward}} \tag{A34}$$

Total hierarchical cost:

$$C_{\text{hierarchical}} = C_{\text{task}} + C_{\text{resource}} + C_{\text{time}} = O(m+n+T) \cdot \mathcal{C}_{\text{forward}} \tag{A35}$$

□

References

1. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]
2. Cheng, R.; Gen, M.; Tsujimura, Y. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation. *Comput. Ind. Eng.* **1996**, *30*, 983–997. [CrossRef]
3. Vinyals, O.; Fortunato, M.; Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems, Proceedings of the 29th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015*; MIT Press: Cambridge, MA, USA, 2015; Volume 28.
4. Kool, W.; Van Hoof, H.; Welling, M. Attention, learn to solve routing problems! In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
5. Khalil, E.; Dai, H.; Zhang, Y.; Dilkina, B.; Song, L. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems, Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2017; Volume 30.
6. Ma, Q.; Ge, S.; He, D.; Thaker, D.; Drori, I. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019*; AAAI: Washington, DC, USA, 2019; Volume 33, pp. 4694–4701.
7. Hottung, A.; Tierney, K. Neural large neighborhood search for the capacitated vehicle routing problem. In *Proceedings of the ECAI 2020—24th European Conference on Artificial Intelligence, Santiago de Compostela, Spain, 29 August–8 September 2020*; IOS Press: Amsterdam, The Netherlands, 2020; pp. 443–450.

8. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural combinatorial optimization with reinforcement learning. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
9. Nazari, M.; Oroojlooy, A.; Snyder, L.; Takác, M. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems, Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018*; Curran Associates Inc.: Red Hook, NY, USA, 2018; Volume 31.
10. Cappart, Q.; Chételat, D.; Khalil, E.B.; Lodi, A.; Morris, C.; Velickovic, P. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–61.
11. Bengio, Y.; Lodi, A.; Prouvost, A. Machine learning for combinatorial optimization: A methodological tour d'horizon. *Eur. J. Oper. Res.* **2021**, *290*, 405–421. [CrossRef]
12. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
13. Panwalkar, S.S.; Iskander, W. A survey of scheduling rules. *Oper. Res.* **1977**, *25*, 45–61. [CrossRef]
14. Blackstone, J.H.; Phillips, D.T.; Hogg, G.L. A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Int. J. Prod. Res.* **1982**, *20*, 27–45. [CrossRef]
15. Pinedo, M.L. *Scheduling: Theory, Algorithms, and Systems*, 5th ed.; Springer: Berlin/Heidelberg, Germany, 2016.
16. Baptiste, P.; Le Pape, C.; Nuijten, W. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
17. Van Laarhoven, P.J.; Aarts, E.H.; Lenstra, J.K. Job shop scheduling by simulated annealing. *Oper. Res.* **1992**, *40*, 113–125. [CrossRef]
18. Shiue, Y.R.; Lee, K.C.; Su, C.T. Real-time scheduling for a smart factory using a reinforcement learning approach. *Comput. Ind. Eng.* **2018**, *125*, 604–614. [CrossRef]
19. Waschneck, B.; Reichstaller, A.; Belzner, L.; Altenmüller, T.; Bauernhansl, T.; Knapp, A.; Kyek, A. Deep reinforcement learning for semiconductor production scheduling. In *Proceedings of the 2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), Saratoga Springs, NY, USA, 30 April–3 May 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 301–306.
20. Zhang, C.; Song, W.; Cao, Z.; Zhang, J.; Tan, P.S.; Chi, X. Learning to dispatch for job shop scheduling via deep reinforcement learning. In *Advances in Neural Information Processing Systems, Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020*; Curran Associates Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1621–1632.
21. Liu, C.L.; Chang, C.C.; Tseng, C.J. Actor-critic deep reinforcement learning for solving job shop scheduling problems. *IEEE Access* **2020**, *8*, 71752–71762. [CrossRef]
22. Bertsimas, D.J.; Simchi-Levi, D. A Vehicle Routing Problem with Stochastic Demand. *Oper. Res.* **1992**, *40*, 674–688. [CrossRef]
23. Chen, X.; Tian, Y. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems, Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019*; Curran Associates Inc.: Red Hook, NY, USA, 2019; Volume 32.
24. Manchanda, S.; Mittal, S.; Dhawan, A.; Medya, S.; Ranu, S.; Singh, A. GCOMB: Learning budget-constrained combinatorial algorithms over billion-sized graphs. In *Advances in Neural Information Processing Systems, Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020*; Curran Associates Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 20000–20011.
25. Peng, B.; Wang, J.; Zhang, Z. A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems. In *Proceedings of the International Symposium on Intelligence Computation and Applications, Guangzhou, China, 16–17 November 2019*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 636–650.
26. Tassel, P.; Gebser, M.; Schekotihin, K. A reinforcement learning environment for job-shop scheduling. In Proceedings of the ICAPS 2021 Workshop on Planning and Reinforcement Learning, Online, 4–6 August 2021.
27. Song, W.; Chen, X.; Li, Q.; Cao, Z. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Trans. Ind. Inform.* **2023**, *19*, 1600–1610. [CrossRef]
28. Park, J.; Chun, J.; Kim, S.H.; Kim, Y.; Park, J. Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning. *Int. J. Prod. Res.* **2021**, *59*, 3360–3377. [CrossRef]
29. Lei, K.; Guo, P.; Zhao, W.; Wang, Y.; Qian, L.; Meng, X.; Tang, L. Solve routing problems with a residual edge-graph attention neural network. *Neurocomputing* **2022**, *508*, 79–98. [CrossRef]
30. Kwon, Y.D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; Min, S. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems, Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020*; Curran Associates Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 21188–21198.
31. Kool, W.; van Hoof, H.; Gromicho, J.; Welling, M. Deep policy dynamic programming for vehicle routing problems. In *Proceedings of the Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Los Angeles, CA, USA, 20–23 June 2022*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 190–213.

32. Feng, Y.; You, H.; Zhang, Z.; Ji, R.; Gao, Y. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 29–31 January 2019*; AAAI: Washington, DC, USA, 2019; Volume 33, pp. 3558–3565.
33. Gao, Y.; Feng, Y.; Ji, S.; Ji, R. HGNN+: General hypergraph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *45*, 3181–3199. [CrossRef]
34. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017*; PMLR: Cambridge, MA, USA, 2017; pp. 1126–1135.
35. Nichol, A.; Achiam, J.; Schulman, J. On first-order meta-learning algorithms. *arXiv* **2018**, arXiv:1803.02999.
36. Antoniou, A.; Edwards, H.; Storkey, A. How to train your MAML. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
37. Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P.L.; Sutskever, I.; Abbeel, P. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv* **2016**, arXiv:1611.02779. [CrossRef]
38. Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019*; PMLR: Cambridge, MA, USA, 2019; pp. 5331–5340.
39. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
40. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [CrossRef]
41. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 7167–7176.
42. Ruder, S.; Peters, M.E.; Swayamdipta, S.; Wolf, T. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, Minneapolis, MN, USA, 3–7 June 2019*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 15–18.
43. Mazyavkina, N.; Sviridov, S.; Ivanov, S.; Burnaev, E. Reinforcement Learning for Combinatorial Optimization: A Survey. *Comput. Oper. Res.* **2021**, *134*, 105400. [CrossRef]
44. Li, Z.; Chen, Q.; Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems, Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018*; Curran Associates Inc.: Red Hook, NY, USA, 2018; Volume 31.
45. Peng, Y.; Choi, B.; Xu, J. Graph learning for combinatorial optimization: A survey of state-of-the-art. *Data Sci. Eng.* **2021**, *6*, 119–141. [CrossRef]
46. Hospedales, T.; Antoniou, A.; Micaelli, P.; Storkey, A. Meta-learning in neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 5149–5169. [CrossRef]
47. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009*; Association for Computing Machinery: New York, NY, USA, 2009; pp. 41–48.
48. Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M.E.; Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *J. Mach. Learn. Res.* **2020**, *21*, 1–50.
49. Altman, E. *Constrained Markov Decision Processes*; CRC Press: Boca Raton, FL, USA, 1999.
50. Achiam, J.; Held, D.; Tamar, A.; Abbeel, P. Constrained policy optimization. In *Proceedings of the International Conference on Machine Learning. PMLR, Sydney, Australia, 6–11 August 2017*; PMLR: Cambridge, MA, USA, 2017; pp. 22–31.
51. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; Van Den Berg, R.; Titov, I.; Welling, M. Modeling relational data with graph convolutional networks. In *Proceedings of the European Semantic Web Conference, Heraklion, Greece, 3–7 June 2018*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 593–607.
52. Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; Yu, P.S. Heterogeneous graph attention network. In *Proceedings of the The World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019*; ACM: New York, NY, USA, 2019; pp. 2022–2032.
53. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*. [CrossRef]
54. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015*; PMLR: Cambridge, MA, USA, 2015; pp. 448–456.
55. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450. [CrossRef]
56. Bacon, P.L.; Harb, J.; Precup, D. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017*; AAAI: Washington, DC, USA, 2017; Volume 31, pp. 1726–1734.
57. Nachum, O.; Gu, S.S.; Lee, H.; Levine, S. Data-efficient hierarchical reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2018**, *31*. [CrossRef]
58. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv* **2015**, arXiv:1508.04025. [CrossRef]

59. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
60. Chow, Y.; Tamar, A.; Mannor, S.; Pavone, M. Risk-sensitive and robust decision-making: A CVaR optimization approach. In *Advances in Neural Information Processing Systems, Proceedings of the 29th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015*; MIT Press: Cambridge, MA, USA, 2015; Volume 28, pp. 1522–1530.
61. Tamar, A.; Chow, Y.; Ghavamzadeh, M.; Mannor, S. Policy gradients with variance related risk criteria. In *Proceedings of the International Conference on Machine Learning. PMLR, Lille, France, 6–11 July 2015*; PMLR: Cambridge, MA, USA, 2015; pp. 387–396.
62. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 618–626.
63. Sundararajan, M.; Taly, A.; Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning. PMLR, Sydney, Australia, 6–11 August 2017*; PMLR: Cambridge, MA, USA, 2017; pp. 3319–3328.
64. Koza, J.R. *Genetic Programming II: Automatic Discovery of Reusable Programs*; MIT Press: Cambridge, MA, USA, 1994.
65. Udrescu, S.M.; Tegmark, M. AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.* **2020**, *6*, eaay2631. [CrossRef] [PubMed]
66. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
67. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.