

Hierarchical Reinforcement Learning for Fleet Control in Multilayer Container Storage Systems

Mike Wesselhöft¹,
Jana Jost¹,
Fabian Menebröcker¹,
Carlos Jahn²

¹Institute of Logistics Engineering,
²Institute of Maritime Logistics,
Hamburg University of Technology
Hamburg, Germany

This work proposes a hierarchical reinforcement learning (HRL) framework for centralised fleet control in a multi-layer container storage system. A simulation environment is developed in NVIDIA Isaac Sim, modelling a multi-floor warehouse with AGV navigation layers and container storage layers connected via lifts. The control architecture is structured into multiple hierarchical levels: (i) a low-level grid controller responsible for local navigation, (ii) a mid-level controller for task and resource allocation, and (iii) a global controller addressing storage optimisation and container rearrangement. The mid-level policy combines a Semi-Markov Decision Process formulation with an entity-based Transformer architecture that jointly reasons over agents, tasks and shared resources. Training is conducted sequentially using curriculum learning. Initial experiments focus on validating the environment and learning feasibility at the individual hierarchy levels; on the mid level, five coordination architectures are compared and reach near-perfect task completion, with a two-stage refinement variant delivering the best overall efficiency. Higher-level storage optimisation remains ongoing work. The proposed framework establishes a foundation for jointly solving routing, scheduling and rearrangement within a unified learning-based paradigm.

[Keywords: hierarchical reinforcement learning, multi-agent systems, automated guided vehicles, container logistics, curriculum learning]

1 Introduction

The increasing demands in modern logistic systems have led to a growing adoption of automated guided vehicles (AGVs) for tasks such as order picking, production supply and container transport [1]. While single-vehicle systems mainly involve infrastructure- and vehicle-level challenges, large-scale fleets require the joint coordination of routing, scheduling and storage management, where individual agent decisions directly influence overall system performance [2].

These challenges are particularly pronounced in container logistics and port environments. Compared to structured indoor warehouses, outdoor systems face additional constraints such as unreliable communication, reduced localisation accuracy and large physical container dimensions. Consequently, many established intralogistic solutions, including top-loading storage systems such as AutoStore, are not directly applicable, which adds additional operational dependencies.

This type of container handling results in rearrangement problems closely related to sliding-tile and token-swapping problems, where blocked containers must first be relocated before transport can occur. As a result, routing, scheduling and rearrangement become tightly coupled optimisation problems, each of which is NP-hard in general [3, 4, 5]. Existing approaches commonly address these sub-problems separately, particularly scheduling and path planning, despite their strong interdependencies, which can limit coordination across decision layers and reduce overall system-level optimisation potential [6, 7, 8].

Reinforcement learning (RL) has emerged as a promising framework for complex sequential decision-making problems [9, 10]. In particular, hierarchical reinforcement

learning (HRL) enables scalable control through multiple levels of abstraction and naturally supports multi-objective optimisation, including criteria such as efficiency and robustness.

In this work, we propose an HRL framework for fleet control in multi-layer container storage systems. The approach combines a Semi-Markov Decision Process (SMDP) formulation with an entity-based Transformer architecture to model interactions between agents, tasks and shared resources. By tackling the problem in multiple control levels, the framework jointly addresses routing, scheduling and rearrangement within a unified learning-based paradigm. The remainder of this paper is organised as follows. Section 2 reviews related work. Sections 3 and 4 present the system architecture and problem formulation. Section 5 introduces the proposed learning approach, Section 6 describes the training setup, Section 7 evaluates its performance and Section 8 concludes the paper.

2 Related Work

Multi-agent reinforcement learning (MARL) has been widely studied for decentralised decision making in cooperative systems. Centralised training with decentralised execution (CTDE) has emerged as a dominant paradigm, enabling agents to condition their policies on global information during training while maintaining decentralised execution [11]. Approaches such as value decomposition networks (VDN) [12] and related value factorisation methods address scalability by decomposing joint value functions, but often struggle with large combinatorial action spaces and structured environments.

The task-assignment problem considered in this work is closely related to combinatorial optimisation and multi-agent path finding (MAPF). MAPF has been extensively studied, with classical approaches focusing on optimal and heuristic search methods such as Conflict-Based Search [13], as well as comprehensive overviews of problem variants and benchmarks [14]. In parallel, multi-robot task allocation is often formulated as a combinatorial assignment problem, frequently solved via auction-based or bipartite matching approaches [15]. Extensions to warehouse and logistic environments integrate task allocation with path planning, but typically rely on handcrafted heuristics and centralised solvers, limiting scalability in dynamic and large-scale settings.

Recent work has applied RL to large-scale resource allocation [16], logistics, task assignment, routing and fleet coordination, demonstrating improved adaptability over rule-based and heuristic methods [17, 18]. MAPF approaches such as PRIMAL further demonstrate the potential of RL for decentralised coordination [10]. However, many existing methods rely on flat or grid-based state representations

and do not explicitly model the relational structure between heterogeneous entities such as agents, tasks and shared resources.

Transformer architectures and attention mechanisms have become increasingly popular for structured decision-making. Set-based models such as the Set Transformer [19] and attention-based routing solvers [20, 21] enable permutation-invariant processing of variable-sized inputs, while attention mechanisms have also been used to improve multi-agent coordination by explicitly modelling agent interactions [22]. These approaches provide a natural framework for representing heterogeneous entities and their relationships.

HRL improves scalability and sample efficiency by decomposing complex problems into multiple levels of abstraction [23]. Frameworks such as the options framework [24] formalise temporal abstraction, while approaches such as FeUdal Networks [25] demonstrate the effectiveness of hierarchical policies for learning structured behaviours. Extensions to multi-agent systems separate high-level coordination from low-level control, making HRL particularly suitable for large-scale logistic problems [26].

In contrast to prior work, this paper combines (i) an SMDP formulation aligned with event-driven decision making, (ii) an entity-based Transformer architecture for relational reasoning and (iii) a hierarchical decomposition tailored to multi-layer warehouse systems, enabling scalable coordination under strong coupling between task assignment, spatial constraints and shared resources.

3 System Architecture

The system operates in a discrete multi-layer warehouse environment composed of vertically stacked grid structures. Each floor is represented as a two-dimensional grid (one agent-navigation layer and one container-storage layer) with heterogeneous cell types including traversable cells, blocked storage locations and designated elevator-access points (Fig. 1). Floors are interconnected via a set of shared elevators with limited capacity, introducing vertical transport constraints and resource contention.

Agents operate on the navigation layer and execute pickup-and-delivery tasks defined by source–destination pairs that may span multiple floors. The environment further incorporates operational constraints such as limited battery capacity and charging stations, requiring agents to balance task execution with energy management. The global system state therefore combines spatial occupancy, agent states, task distributions and resource availability, forming a structured, high-dimensional decision space. The simulation is implemented in NVIDIA Isaac Sim, which pro-

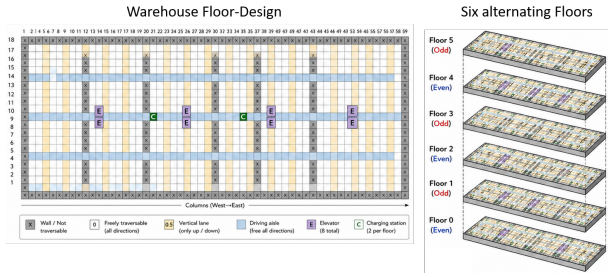


Figure 1: Warehouse layout used throughout the experiments.

vides GPU-accelerated multi-environment rollouts required for large-scale RL training.

The proposed system follows a three-tier hierarchical control architecture (Fig. 2) in which each level operates at a different level of abstraction. The lowest level is responsible for motion execution and handles geometric navigation on the discrete grid. The mid-level controller represents the core learning component of the system. It operates at a higher temporal abstraction and is responsible for assigning tasks to agents and coordinating shared resources such as elevators. Rather than controlling individual movements, it issues high-level decisions that are subsequently executed by the low-level controller. The top-level controller, which is currently under development, is intended to manage global storage strategies and to solve rearrangement problems arising from blocked containers. This hierarchical decomposition enables a clear separation of concerns while still allowing interactions between levels through shared objectives and reward signals.

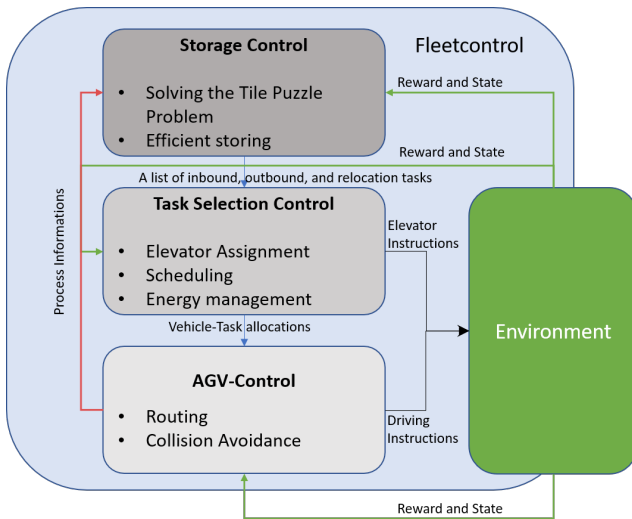


Figure 2: Hierarchical reinforcement learning based control architecture.

4 Problem Formulation

The mid-level controller is modelled as an SMDP, which is more suitable than a standard Markov Decision Process due to the event-driven nature of the problem. Decisions are not required at every simulation step but only when significant events occur, such as task completion or agent idleness. This formulation reduces the number of ineffective updates and aligns the learning process with meaningful decision points.

We denote the SMDP tuple as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \tau)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $\mathcal{P}$, reward function R , discount factor γ and dwell-time distribution τ . A state $\mathbf{s}_t \in \mathcal{S}$ groups four typed subsets: per-agent features $\mathbf{X}^{\text{ag}} \in \mathbb{R}^{N_{\text{max}} \times d_{\text{ag}}}$ (position, current phase, battery level, currently assigned task index, elevator sub-state), per-task features $\mathbf{X}^{\text{ts}} \in \mathbb{R}^{K \times d_{\text{ts}}}$ (source/target cell, floor pair, urgency, assignment flag), a per-floor navigation grid $\mathbf{G} \in \{0, 1\}^{L \times H \times W \times 2}$ (occupancy and container layers), and per-elevator features $\mathbf{X}^{\text{el}} \in \mathbb{R}^{E \times d_{\text{el}}}$ (cabin floor, queue length, door sub-state).

The action at each decision is a joint per-agent assignment $\mathbf{a}_t = (a_t^1, \dots, a_t^N)$, where each $a_t^i = (a_t^{i, \text{tsk}}, a_t^{i, \text{elv}})$ pairs a task selection $a_t^{i, \text{tsk}} \in \{0, \dots, K+4\}$ (with special IDs for IDLE, WAIT, CHARGE, EMERGENCY_DROP and STAY) with an elevator assignment $a_t^{i, \text{elv}} \in \{0, \dots, E, \text{NONE}\}$. The full joint action space is combinatorial in N and K ; tractability is achieved via factorisation and state-conditioned action masking (Section 5).

Between two decision events, K simulation steps elapse in the low-level simulator; the SMDP return is

$$G_t = \sum_{k=0}^{K-1} \gamma^k R_{t+k}, \quad (1)$$

which folds the accumulated dense signal into a single transition $(\mathbf{s}_t, \mathbf{a}_t, G_t, \mathbf{s}_{t+K})$ and improves the signal-to-noise ratio during training. The optimisation objective is the standard expected discounted return

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t R(\mathbf{s}_t, \mathbf{a}_t) \right], \quad (2)$$

with policy π_θ and episode length T . The concrete reward R that instantiates this objective for our warehouse setting is defined in Section 6.

5 Learning Approach

This section describes the proposed hierarchical learning approach. It first presents low-level navigation, followed by the mid-level coordination policies investigated in this work.

5.1 Low-Level Navigation

We evaluated several RL algorithms on an isolated single-agent navigation task using the same eight-stage curriculum employed throughout this work. The results are summarised in Table 1. Although the RL baselines achieve high success rates, none outperform a classical A* planner, which computes optimal paths in milliseconds and solves all curriculum levels by construction [27]. Since low-level navigation is a deterministic shortest-path problem, we employ A* at the lowest layer and focus RL on higher-level coordination, resource contention and combinatorial assignment. This choice improves interpretability and reduces training complexity.

Table 1: RL baselines on the isolated low-level navigation task.

Algorithm	Success Rate	Avg. Steps	Curriculum
A2C	97 %	143	8/8
PPO (CNN)	96 %	103	8/8
DQN	98 %	56	4/8
MaskablePPO	95 %	100	8/8

5.2 Mid-Level Policy and Coordination

The mid-level controller is modelled as a centralised stochastic policy

$$\pi_{\theta}(\mathbf{a}_t \mid \mathbf{s}_t) \quad (3)$$

with per-agent factorised actions $\mathbf{a}_t = (a_t^1, \dots, a_t^N)$ and $\mathbf{s}_t$ the current environment state. The global state is represented as a set of typed entities (agents, tasks, elevators and grid cells). Agent, task and elevator entities are projected into a shared latent space by type-specific linear embeddings. The multi-floor grid, in contrast, is encoded by a small convolutional network applied independently to each floor: three strided 3×3 convolutions ($2 \rightarrow 32 \rightarrow 64 \rightarrow 128$ channels, each with batch normalisation and ReLU) followed by global average pooling and a linear projection produce one d -dimensional token per floor. This per-floor CNN encoder is used by the GRU, Transformer, Auction Transformer and Double Transformer variants; the MAT-Sparse variant replaces it with a sparse set of cell tokens that bypass the CNN entirely. All resulting tokens (entities and grid alike) are then processed jointly by a Transformer encoder [28], and per-agent action heads produce task and elevator assignments from the corresponding agent embeddings. This entity-based formulation is permutation-invariant over each entity type and scales naturally with the number of agents and tasks.

5.3 Feasibility via Action Masking

The set of valid actions varies with the system state (e.g. assigned tasks, occupied elevators, depleted batteries). Action masking is applied to the policy logits so that only feasi-

ble assignments are sampled, which simplifies exploration and prevents the policy from spending capacity on encoding hard constraints.

5.4 Coordination Variants

We compare five mid-level architectures, each implementing a different coordination mechanism on top of the same entity-based backbone:

- **GRU** A recurrent baseline that processes a flattened state vector with a gated recurrent unit [29] and produces per-agent logits from the final hidden state. Coordination is purely implicit through the shared recurrent state.
- **Transformer** The entity-based Transformer described above. Coordination emerges from self-attention across all entities, without a conflict-resolution mechanism.
- **Auction Transformer (AT)** Extends the Transformer policy with a greedy auction over task logits at sampling time: agents are ranked by logit magnitude and assigned tasks sequentially, masking each claimed task before the next assignment. This guarantees duplicate-free assignments and avoids the gradient mismatch that arises when the environment silently reassigns conflicting actions.
- **Double Transformer (DT)** A two-stage policy in which a first Transformer produces draft action distributions $\mathbf{p}^{(1)}$ and a second Transformer refines per-agent decisions based on these drafts, $\pi_{\theta}^{(2)}(a_i \mid \mathbf{s}_t, \mathbf{p}^{(1)})$. This allows each agent to react to the predicted intentions of others.
- **MAT-Sparse** A multi-agent Transformer based on Wen *et al.* [30], which decodes agent actions sequentially such that each agent conditions on the embeddings and previously selected actions of others. We use a sparse cell-token representation to keep the input size independent of grid resolution.

All five architectures share the same observation and action spaces, masking logic and reward function; only the policy network and its coordination mechanism differ. This isolates the effect of the coordination strategy from the rest of the pipeline.

6 Training Setup

6.1 Reward Function

The mid-level reward decomposes into three groups: a dense per-decision term R^{step} used for credit assignment, sparse event bonuses R^{evt} triggered by task completions and failure

modes, and a terminal bonus R_t^{term} paid once the episode ends. All time-dependent components are normalised by the episode task budget x so that the expected return remains on the same scale across curriculum levels with different task loads. The total reward at decision t is

$$R_t = R_t^{\text{step}} + R_t^{\text{evt}} + R_t^{\text{term}}. \quad (4)$$

The dense term aggregates busy-agent effort, low-battery pressure and elevator contention,

$$R_t^{\text{step}} = \frac{1}{x} \left(w_s n_t^{\text{busy}} + w_{\text{lb}} n_t^{\text{low}} + w_{\text{ec}} n_t^{\text{ec}} \right), \quad (5)$$

where n_t^{busy} counts non-idle agents, n_t^{low} counts agents below the low-battery threshold and n_t^{ec} counts excess agents queued at a shared elevator. The event term collects instantaneous outcomes,

$$R_t^{\text{evt}} = w_{\text{ic}} \Delta n_t^{\text{done}} + w_{\text{eff}} \frac{1}{x} e_t + w_{\text{st}} \mathbf{1}[\text{stuck}] + w_{\text{ed}} \mathbf{1}[\text{drop}], \quad (6)$$

with Δn_t^{done} the number of tasks completed at step t and e_t a per-task efficiency bonus that rewards near-optimal execution time. The terminal component fires exactly once,

$$R_T^{\text{term}} = \begin{cases} w_{\text{ep}}, & \text{all tasks completed,} \\ w_{\text{to}}, & \text{episode timeout,} \\ w_{\text{db}}, & \text{dead-battery failure.} \end{cases} \quad (7)$$

The numerical weights used throughout training are listed in Table 2. They were kept identical across all five coordination architectures so that the coordination mechanism is the only free variable in the comparison.

Table 2: Mid-level reward weights.

Symbol	Value	Meaning
w_s	-0.2	per busy-agent step penalty
w_{lb}	-0.1	per low-battery agent, per step
w_{ec}	-1.0	excess agent per elevator, per step
w_{ic}	+10.0	task completed (pickup and delivery)
w_{eff}	+2.0	per-task efficiency bonus scale
w_{st}	-10.0	agent flagged stuck
w_{ed}	-10.0	emergency container drop
w_{ep}	+50.0	terminal: full episode success
w_{to}	-150.0	terminal: episode timeout
w_{db}	-150.0	terminal: dead battery

6.2 Algorithm and Hyperparameters

We optimise the policy with Proximal Policy Optimisation using the RL Games implementation and its Isaac-Lab integration on 256 parallel GPU environments. Compared to off-policy value-based methods, on-policy PPO is a robust match for the discrete multi-head action space induced by the joint per-agent assignment: it accommodates action

masking directly at the logit level, its clipped-objective update is stable under the temporally extended SMDP transitions, and its parallel-environment throughput on GPU matches the Isaac-Sim rollout rate. Q-learning-style methods such as QMIX [11] and VDN [12] were considered but discarded for the mid level: they target value-factorisation under decentralised execution, while our mid-level controller is a single centralised policy that already observes the global state and produces the full joint action in one forward pass. Decentralised MARL variants remain a natural next step once the top-level controller distributes decisions across regions.

Hyperparameters were selected in three stages. Discount, GAE parameter, clip range, gradient clipping and mini-epoch count were inherited from published PPO defaults on Isaac-Lab [9]. Rollout, batch and minibatch sizes were tuned via a coarse grid on the smallest curriculum level (level 0) to balance FPS and sample reuse. Learning rate and entropy coefficient were then swept in the range $\{1\text{e-}4, 2\text{e-}4, 3\text{e-}4\}$ and $\{0.02, 0.05, 0.08\}$ respectively; the chosen combination attained the fastest promotion through the first three curriculum levels. An adaptive KL-controlled learning-rate schedule (threshold 8×10^{-3}) then handles the remainder of training. The same values are shared by all five coordination architectures. Table 3 lists the final settings.

Table 3: PPO training hyperparameters.

Hyperparameter	Value
Discount factor γ	0.99
GAE parameter λ	0.95
Clip range ϵ	0.2
Value clip	enabled
Learning rate	2×10^{-4} (adaptive)
KL threshold	8×10^{-3}
Entropy coefficient	0.08
Value-loss coefficient	2.0
Mini-epochs per update	6
Rollout horizon (per env)	1024
Minibatch size	4096
Parallel environments	256
Gradient clip (L2 norm)	1.0
Optimiser	Adam
Observation / value normalisation	enabled

6.3 Curriculum

Training uses a nine-stage staircase curriculum that alternates between adding a floor and adding agents, so the policy first masters spatial routing before the coordination complexity grows (Table 4). Each stage fixes the number of agents, the number of accessible floors, the total task budget per episode and the probability that a task spans two floors. A stage advances to the next stage once two criteria are jointly met over the last 100 episodes: (i) a task-completion

rate of 100% (episodes end with all tasks delivered and no dead-battery event) and (ii) an average steps-per-task ratio below $\text{slack}_\ell \cdot t^*$, where t^* is an analytical lower bound on the optimal execution time and slack_ℓ shrinks from 1.50 at stage 0 to 1.18 at stage 8. Two guardrails stabilise the schedule: a 2000-episode warmup during which no promotion is granted, and a 200-episode grace window at each newly entered stage. If the completion rate drops below 50% for 200 episodes at the new stage, the policy is demoted one level to prevent catastrophic forgetting [31].

Table 4: Nine-stage staircase curriculum.

Stage	Agents	Floors	Tasks	Cross-Floor
0	5	2	10	20 %
1	5	3	10	24 %
2	8	3	10	28 %
3	8	4	10–15	32 %
4	10	4	12–18	36 %
5	10	5	14–22	40 %
6	12	5	16–26	44 %
7	12	6	18–30	48 %
8	15	6	20–35	50 %

6.4 Reproducibility

Experiments were run on a workstation with a single NVIDIA GPU using Isaac Sim 4.5 and Isaac Lab as the simulation backend; RL Games provides the PPO trainer. The simulator runs at 60 Hz with an episode budget of 20 s (≈ 1200 low-level steps). The environment declares 20 agent slots, a task queue of size 20, up to 6 floors and 8 elevators; each floor is a 59×18 grid (see Fig. 1) with fixed elevator positions and per-elevator waiting cells. The battery model drains 0.2%/movement, recharges 2%/step at charging stations and uses a soft-mask threshold of 20 % below which only IDLE, WAIT and CHARGE remain in the action mask. All reported experiments use a fixed seed of 42; Section 7 discusses multi-seed generalisation as ongoing work. Task generation is A*-validated at episode start so that every episode is provably solvable; a small 1 % fraction of intentionally unsolvable tasks is injected as robustness noise. Convergence is declared when the smoothed task-completion rate at the final curriculum stage remains $\geq 99\%$ for 500 consecutive episodes and the average episode length ceases to decrease by more than 1 % per 100 episodes.

7 Experiments

All variants are evaluated on the final curriculum level (15 agents, 6 floors, 20–35 tasks per episode) over the most recent 100 episodes of training. Reported metrics are the success rate (episodes with all tasks completed and no dead-battery failure), the average episode length in simulation

steps and the average number of simulation steps per completed task.

Table 5: Mid-level coordination architectures – performance at the final curriculum level.

Architecture	Success Rate	Avg. Steps	Steps / Task
GRU	99.6 %	204	7.4
Transformer	99.5 %	226	8.2
AT	99.6 %	191	7.0
DT	99.8 %	184	6.8
MAT-Sparse	99.3 %	168	6.3

All architectures achieve near-perfect task completion at the final curriculum level (Table 5), demonstrating that the entity-based observation, action masking and SMDP formulation provide a solvable learning framework. Success rates range from 99.3 % to 99.8 %, indicating that all architectures learn effective fleet coordination policies. The primary differences lie in execution efficiency rather than task completion.

The DT achieves the best overall results, obtaining the highest success rate (99.8 %), the shortest episodes (184 steps) and a steps-per-task value of 6.8. The MAT-Sparse variant reaches the joint-lowest steps-per-task (6.3) with episode length 168 steps, while the AT sits between them (191 steps, 7.0 steps/task). In contrast, the GRU and plain Transformer require more steps to complete the same workload, with the plain Transformer showing the highest average episode length (226 steps) and steps per task (8.2). Overall, the results show that while task completion is largely determined by the proposed framework, the choice of coordination architecture remains important for execution efficiency, leading to measurable differences in fleet performance.

7.1 Ablation of the coordination mechanism

Table 5 can be read as a controlled ablation of the coordination mechanism: the observation and action spaces, action masking, SMDP formulation, reward function and PPO hyperparameters (Section 6) are held fixed across all five rows; only the policy head and its coordination strategy change. Under this design the near-identical success rates confirm that the proposed framework itself, and not any single coordination architecture, is responsible for reaching task completion, while the 8 % efficiency gap between the plain Transformer and the DT isolates the effect of explicit intention-refinement. Independent ablations of the individual framework components (action masking, SMDP aggregation, curriculum staircase) share the same PPO backbone and are natural extensions; we defer them so that this paper stays focused on the coordination comparison.

7.2 Scalability

The final curriculum stage already stresses the mid level with 15 agents, 6 floors and up to 35 tasks per episode, giving a joint action space of $\mathcal{O}((K+5)^N \cdot (E+1)^N)$. Two mechanisms keep the wall-clock cost tractable. First, the policy factorises the joint action into per-agent heads so that the sampled action grows linearly in N rather than combinatorially. Second, all coordination variants operate over an entity-token set whose size is fixed by the observation-space dimensions (20 agent, 20 task, 6 floor, 8 elevator tokens), so a single forward pass is independent of the actual number of active agents or tasks. Inference-time cost is dominated by the Transformer encoder and remains sub-millisecond per decision on a single GPU for all variants, well below the SMDP decision cadence. Scaling beyond the current bounds (more agents or floors) would require enlarging the token budget and re-tuning the curriculum but leaves the algorithmic pipeline unchanged.

7.3 Reference baselines and seeds

Because our primary aim is to compare coordination architectures under identical training conditions, all reported numbers correspond to a single fixed seed and no rule-based reference is included in Table 5. Two extensions are planned. First, repeating each configuration with additional random seeds so that mean and standard deviation can be reported. Second, adding a greedy nearest-task heuristic and a random-assignment baseline as performance floors. Preliminary internal runs of the greedy heuristic reach the initial curriculum stages but stall on the higher levels due to elevator contention and cross-floor task routing, consistent with the difficulty ranking observed for the learned policies.

8 Conclusion

We presented an HRL framework for fleet control in multi-layer container storage systems. Across all evaluated architectures, the proposed entity-based observation, action masking and SMDP formulation achieve near-perfect task completion under realistic battery and elevator constraints. Among the tested coordination architectures, the DT delivers the best overall performance with a 99.8% success rate, the shortest episodes and near-lowest execution cost per task. The AT and MAT-Sparse variants achieve similarly strong results, indicating that multiple coordination mechanisms can be effectively integrated into the proposed framework.

Future work addresses the global rearrangement level and large-scale deployment in mixed indoor/outdoor container environments, together with a refinement of the reward formulation, including incentive-based coordination mechanisms inspired by game-theoretic approaches [32]. Integration of the top-level storage controller will follow

the same event-driven SMDP contract used at the mid level: high-level decisions issue container rearrangement targets that are consumed as task streams by the mid-level policy, which continues to allocate agents and elevator slots. The two levels share the observation backbone but operate on disjoint action spaces, so the mid-level policy trained here can be reused unchanged as a lower-layer executor. On the evaluation side, planned extensions include multi-seed runs with reported mean and standard deviation, rule-based reference baselines (nearest-task greedy and random assignment) and additional system-level metrics such as elevator waiting time and effective AGV utilisation.

REFERENCES

- [1] International Federation of Robotics, “Robot sales rise again,” Frankfurt am Main, Germany, 2021, press release / report. [Online]. Available: <https://ifr.org/ifr-press-releases/news/robot-sales-rise-again>
- [2] W. Lu, S. Guo, T. Song, and Y. Li, “Analysis of multi-AGVs management system and key issues: A review,” *Computer Modeling in Engineering & Sciences*, vol. 131, no. 3, pp. 1197–1227, 2022. [Online]. Available: <https://doi.org/10.32604/cmes.2022.019770>
- [3] J. Yu and S. M. LaValle, “Structure and intractability of optimal multi-robot path planning on graphs,” in *Proc. AAAI Conf. on Artificial Intelligence*, vol. 27, no. 1, 2013, pp. 1443–1449. [Online]. Available: <https://arxiv.org/abs/1301.5977>
- [4] M. Pinedo and K. Hadavi, “Scheduling: Theory, algorithms and systems development,” in *Operations Research Proceedings 1991*. Springer, 1992, pp. 35–42.
- [5] D. Ratner and M. Warmuth, “Finding a shortest solution for the $N \times N$ extension of the 15-puzzle is intractable,” in *Proc. AAAI Nat. Conf. on Artificial Intelligence*, 1986, pp. 168–172.
- [6] S. Al Mahmud, A. M. Ibrahim, and M. R. Khan, “A comprehensive review on multi-robot task assignment and path planning: operational constraints, challenges and potentials of deep reinforcement learning,” *Applied Soft Computing*, 2025. [Online]. Available: <https://doi.org/10.1016/j.asoc.2025.114498>
- [7] C. Chen, Y. Wang, X. Chang, and J. Shi, “Integrated task assignment and path planning for collision-minimizing multiple Automated guided vehicles scheduling,” *Expert Systems with Applications*, vol. 302, p. 130615, 2026. [Online]. Available: <https://doi.org/10.1016/j.eswa.2025.130615>
- [8] M. Wesselhöft, J. Hinckeldeyn, and J. Kreutzfeldt, “Controlling fleets of autonomous mobile robots with reinforcement learning: A brief survey,” *Robotics*, vol. 11, no. 5, p. 85, 2022. [Online]. Available: <https://doi.org/10.3390/robotics11050085>

- [9] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, pp. 350–354, 2019. [Online]. Available: <https://doi.org/10.1038/s41586-019-1724-z>
- [10] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, “PRIMAL2: Pathfinding via reinforcement and imitation multi-agent learning—lifelong,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, 2021. [Online]. Available: <https://doi.org/10.1109/LRA.2021.3062803>
- [11] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic value function factorisation for deep multi-agent reinforcement learning,” *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020. [Online]. Available: <https://jmlr.org/papers/v21/20-081.html>
- [12] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, “Value-decomposition networks for cooperative multi-agent learning,” in *Proc. 17th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2018. [Online]. Available: <https://arxiv.org/abs/1706.05296>
- [13] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 219, pp. 40–66, 2015. [Online]. Available: <https://doi.org/10.1016/j.artint.2014.11.006>
- [14] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, R. Barták, and E. Boyarski, “Multi-agent pathfinding: Definitions, variants, and benchmarks,” *AI Magazine*, 2019. [Online]. Available: <https://arxiv.org/abs/1906.08291>
- [15] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *Int. Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004. [Online]. Available: <https://doi.org/10.1177/0278364904045564>
- [16] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, “Large-scale order dispatch in on-demand ride-sharing platforms: A learning and planning approach,” in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (KDD)*, 2018. [Online]. Available: <https://doi.org/10.1145/3219819.3219824>
- [17] K. Lin, R. Zhao, Z. Xu, and J. Zhou, “Efficient large-scale fleet management via multi-agent deep reinforcement learning,” in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (KDD)*, 2018. [Online]. Available: <https://doi.org/10.1145/3219819.3219993>
- [18] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takáč, “Reinforcement learning for solving the vehicle routing problem,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. [Online]. Available: <https://arxiv.org/abs/1802.04240>
- [19] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, “Set transformer: A framework for attention-based permutation-invariant neural networks,” in *Proc. Int. Conf. on Machine Learning (ICML)*, 2019. [Online]. Available: <https://arxiv.org/abs/1810.00825>
- [20] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” in *Int. Conf. on Learning Representations (ICLR) Workshop*, 2017. [Online]. Available: <https://arxiv.org/abs/1611.09940>
- [21] W. Kool, H. van Hoof, and M. Welling, “Attention, learn to solve routing problems!” in *Int. Conf. on Learning Representations (ICLR)*, 2019. [Online]. Available: <https://arxiv.org/abs/1803.08475>
- [22] S. Iqbal and F. Sha, “Actor-attention-critic for multi-agent reinforcement learning,” in *Proc. Int. Conf. on Machine Learning (ICML)*, 2019. [Online]. Available: <https://arxiv.org/abs/1810.02912>
- [23] A. G. Barto and S. Mahadevan, “Recent advances in hierarchical reinforcement learning,” *Discrete Event Dynamic Systems*, vol. 13, no. 1–2, pp. 41–77, 2003. [Online]. Available: <https://doi.org/10.1023/A:1022140919877>
- [24] R. S. Sutton, D. Precup, and S. Singh, “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning,” *Artificial Intelligence*, vol. 112, no. 1–2, pp. 181–211, 1999. [Online]. Available: [https://doi.org/10.1016/S0004-3702\(99\)00052-1](https://doi.org/10.1016/S0004-3702(99)00052-1)
- [25] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, “FeUdal networks for hierarchical reinforcement learning,” in *Proc. Int. Conf. on Machine Learning (ICML)*, 2017. [Online]. Available: <https://arxiv.org/abs/1703.01161>
- [26] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, “Multiagent bidirectionally-coordinated nets for learning to play StarCraft combat games,” in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2017. [Online]. Available: <https://arxiv.org/abs/1703.10069>
- [27] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. [Online]. Available: <https://doi.org/10.1109/TSSC.1968.300136>
- [28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin,

- “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. [Online]. Available: <https://arxiv.org/abs/1706.03762>
- [29] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014. [Online]. Available: <https://arxiv.org/abs/1406.1078>
- [30] M. Wen, J. G. Kuba, R. Lin, W. Zhang, Y. Wen, J. Wang, and Y. Yang, “Multi-agent reinforcement learning is a sequence modeling problem,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. [Online]. Available: <https://arxiv.org/abs/2205.14953>
- [31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proc. 26th Int. Conf. on Machine Learning (ICML)*, 2009, pp. 41–48. [Online]. Available: <https://doi.org/10.1145/1553374.1553380>
- [32] D. Hyland, J. Gutierrez, and M. Wooldridge, “Incentive engineering for concurrent games,” *arXiv preprint arXiv:2307.05076*, 2023. [Online]. Available: <https://arxiv.org/abs/2307.05076>

Mike Wesselhöft, M.Sc., PhD candidate at the Institute of Logistics Engineering, Hamburg University of Technology. His research focuses on reinforcement-learning-based fleet control for automated guided vehicles in container-storage systems.

Address: Institute of Logistical Engineering, Hamburg University of Technology, Theodor-Yorck Straße 8, 21073 Hamburg, Germany,
E-Mail: mike.wesselhoeft@tuhh.de