Hierarchical Deep Reinforcement Learning with Spatiotemporal Encoding for Resilient, Decentralized Multi-Agent Tasking in 2D Environments

Hubert Kyerematen-Boateng, Anthony Herron, Oluwabukunmi David Jaiyeola, Seonho Choi, and Darsana Josyula

Department of Computer Science, Bowie State University, Maryland, USA

Abstract. Multi-agent task allocation under strict temporal constraints poses fundamental challenges. We consider a fully decentralized setting: agents operate on a shared 2D grid, each observing only its own state and assigned task set, and acting in real time without centralized coordination. Hard deadlines impose zero-yield penalties on late harvests, making deadline satisfaction a hard constraint rather than a soft incentive. We present a hierarchical deep reinforcement learning (HDRL) framework that integrates explicit spatiotemporal encoding with a decentralized neural retasking mechanism. Specialized neural encoders capture distance-aware spatial relationships and deadline-sensitive temporal dependencies as first-class architectural components, enabling agents to trade off immediate task value against time criticality and user-defined priorities. A lightweight retasking network performs real-time reassignment of orphaned tasks upon agent failure without centralized coordination. We evaluate the framework on a multi-agent flower-harvesting benchmark featuring soft and hard deadlines, dynamic priority constraints, and a 30% stochastic agent-failure rate, comparing Double Deep Q-Networks (DDQN) and REINFORCE Policy Gradient (PG) against a Mixed-Integer Linear Programming (MILP) optimum. Results reveal a horizon-dependent crossover: DDQN leads at short time budgets ($T \leq 300$ timesteps), while PG surpasses DDQN and exceeds the MILP optimum from $T \geq 400$ onward—reaching 117.8% of the MILP objective at $T = 800$ by recovering from agent failures that the static MILP cannot anticipate. These results establish hierarchical spatiotemporal encoding as a viable approach for real-time multi-agent coordination in robotics, logistics, and autonomous systems.

Keywords: multi-agent reinforcement learning, hierarchical reinforcement learning, spatiotemporal encoding, task allocation, fault tolerance, DDQN

1 Introduction

Multi-agent task allocation (MATA) under temporal constraints is a central challenge in robotics, logistics, and autonomous systems [8, 12]. Consider autonomous drones conducting time-critical deliveries [33] or warehouse robots fulfilling deadline-constrained orders [26]: agents must coordinate to complete spatially distributed tasks with dynamic priorities, soft and hard deadlines, and no guarantee of operational continuity. This problem is characterized by three interlocking difficulties. First, *spatial heterogeneity* means that the cost of reaching a task varies across agents and evolves as the environment changes. Second, *temporal urgency* means that the value of completing a task decays—or collapses to zero—as its deadline approaches. Third, *runtime failures* mean that the task portfolio must be re-distributed on-the-fly without centrally coordinated communication.

Classical approaches expose these tensions sharply. Mixed-Integer Linear Programming (MILP) [5] yields provably optimal allocations under perfect information, but its exponential worst-case complexity renders online replanning intractable in dynamic environments [23]. Standard multi-agent reinforcement learning (MARL) avoids the planning bottleneck but struggles with hard temporal constraints because it optimizes cumulative discounted reward rather than deadline satisfaction [20]; moreover, flat state representations lack the explicit spatial and temporal structure needed for reliable deadline-aware reasoning [31]. The

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp.101-119, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160809

convergence of strict deadlines, spatial task distribution, and runtime failures therefore creates a problem space that resists both classical optimization and conventional RL [3].

We address this gap with a hierarchical deep reinforcement learning framework whose architecture treats space and time as first-class citizens. Unlike prior HRL approaches such as FeUdal Networks [29] and HIRO [18], which communicate goals through unconstrained continuous latent spaces, our coordinator operates over a discrete, semantically interpretable goal set $\mathcal{G}$ with goal-conditioned action masking that enforces hard constraint satisfaction at every timestep. Unlike Centralized Training with Decentralized Execution (CTDE) methods such as QMIX [22] and COMA [6] that require shared value information at training time, our framework shares only the orphan list from failed agents at execution time, making it robust to network failures and naturally applicable to asynchronous deployment. Our contributions are:

1. **Hierarchical spatiotemporal framework.** A two-level coordinator–worker hierarchy in which dedicated spatial and temporal encoders supply distance-aware and deadline-sensitive inductive biases to both Q-value and policy-gradient agents, with goal-conditioned action masking ensuring hard deadline compliance at every decision step.
2. **Multi-agent bee-foraging benchmark.** A configurable 2D grid environment with stochastic agent failures, heterogeneous flower priorities, and mixed soft/hard deadlines that exposes the full difficulty of the problem class.
3. **Neural retasking mechanism.** A decentralized attention-based network that reassigns orphaned tasks in real time upon agent failure, without global coordination, enabling graceful degradation under stochastic failures.
4. **Systematic comparative evaluation.** A rigorous empirical study comparing DDQN and REINFORCE PG against a MILP baseline across temporal horizons of 100–1000 timesteps, revealing a horizon-dependent crossover in which DDQN dominates at short horizons ($T \leq 300$) while PG surpasses the MILP optimum at longer horizons ($T \geq 400$) via learned failure recovery.

The remainder of this paper is organized as follows. Section 2 surveys related work. Section 3 formalizes the environment and problem. Section 4 presents the full architecture and training procedure. Section 5 describes the experimental setup and results. Section 6 analyzes findings and situates them in the broader literature. Section 7 concludes.

2 Related Work

2.1 Multi-Agent Task Allocation Under Temporal Constraints

MATA is formally characterized by Gerkey and Matarić [8] along dimensions of task instantiation, assignment cardinality, and inter-task dependency. Korsah et al. [12] extended this taxonomy to accommodate cross-schedule dependencies and shared resource constraints. MILP [5] remains the standard for offline MATA, yielding optimal solutions under complete information; however, its exponential worst-case complexity renders online replanning intractable [23]. Choudhury et al. [3] showed that state representations encoding deadline urgency directly significantly improve constraint-satisfaction rates over deadline-agnostic methods—a finding that directly motivates our temporal encoder design.

2.2 Deep Q-Networks and the DDQN Correction

Mnih et al. [16] demonstrated that combining Q-learning with deep convolutional function approximation, experience replay [15], and a frozen target network produces human-level

control on Atari benchmarks. Van Hasselt [10] diagnosed the systematic overestimation bias of vanilla Q-learning, and the Double DQN (DDQN) algorithm [27] corrected it by decoupling action selection (online network) from value evaluation (target network). Wang et al. [30] further decomposed Q-values into state-value and action advantage streams, enabling efficient generalization over large, variable-cardinality action spaces—a property directly applicable to our per-agent task sets. These algorithmic advances underpin our coordinator–worker networks.

2.3 Hierarchical Reinforcement Learning

Hierarchical RL decomposes long-horizon problems into a hierarchy of sub-policies operating at distinct temporal and semantic abstractions [13]. Sutton et al. [24] provided the foundational *Options* framework; Vezhnevets et al. [29] operationalized hierarchical decomposition via FeUdal Networks, in which a manager communicates subgoals to a worker in a learned latent space; and Nachum et al. [18] addressed data inefficiency through off-policy correction. In a closely related domain, Cui et al. [4] demonstrated that hierarchical DRL markedly reduces decision complexity in cloud-task scheduling with dynamic action spaces. Our framework applies the same decomposition principle to multi-agent spatial task allocation subject to both soft and hard temporal deadlines.

2.4 Cooperative Multi-Agent Deep Reinforcement Learning

Cooperative MARL has been advanced substantially by the CTDE paradigm [20]. QMIX [22] learns a monotonic mixing network over per-agent utilities; COMA [6] introduces counterfactual baselines to isolate individual agent contributions from shared team rewards; and Foerster et al. [7] showed that agents can learn emergent communication protocols without hand-crafted messaging. Gronauer and Diepold [9] surveyed scalability challenges including non-stationarity and partial observability. In contrast to communication-based cooperative methods, our framework operates under fully decentralized execution with no inter-agent messaging; resilience is instead achieved through the neural retasking module.

2.5 Spatiotemporal Representation Learning

Explicit spatial and temporal reasoning is a key differentiator of high-performance policies in structured multi-agent environments. Wang et al. [31] showed that heterogeneous graph attention networks encoding spatial proximity and deadline constraints outperform flat state representations in scheduling tasks. Jia et al. [11] developed a heterogeneous driving graph transformer achieving state-of-the-art trajectory prediction by encoding scene-level spatial context—conceptually aligned with our multi-scale sinusoidal position embeddings. The Transformer attention mechanism [28] has become the standard architectural primitive for modelling permutation-invariant sets of variable length, which is precisely the structure of our per-agent task sets.

2.6 Fault Tolerance and Priority-Adaptive Policies

Resilience to agent failure and responsiveness to dynamic task priorities are demanding requirements in deployed autonomous systems. El Alami and Rawat [1] applied DQN and policy gradient methods to satellite constellation retasking, where rapid adaptation to component failures is operationally critical; their evaluation is based on average response time and task completion rate. Mao et al. [2] proposed a DRL actor-critic framework

with dynamic task-priority adjustment and demonstrated that priority-adaptive policies outperform fixed-priority baselines—a finding corroborated by our priority-weighted reward shaping. Unlike prior retasking approaches that rely on centralized replanning, our neural retasking module performs real-time task reacquisition through a fully decentralized learned policy, enabling graceful degradation under stochastic failures.

3 System Model and Problem Formulation

3.1 Bee-Foraging Environment

The bee-foraging environment [21] is implemented as a discrete two-dimensional grid-based simulation in which autonomous agents (bees) navigate a grid to harvest pollen from flowers. Formally, the simulation grid is $G \in \mathbb{R}^{g \times g}$ with fixed dimension g. A set of N bee agents is defined as $B = \{b_0, b_1, \ldots, b_{N-1}\}$, where each bee b_i is pre-assigned a subset of target flowers. The flower set is denoted $F = \{f_0, f_1, \ldots, f_{M-1}\}$, where each flower f_j is identified by its index j. At the beginning of each episode, agents (b_i) are assigned a set of flowers that the agents can harvest which is denoted as $\mathcal{F}_{b_i}$. Agents can only harvest flowers that has been assigned to it. Each episode is constrained by a maximum timestep limit T, representing the allocated time horizon.

The grid discretization maps each cell to a unit timestep, establishing a direct correspondence between spatial distance and temporal cost. Agents advance linearly at a uniform velocity of one cell per timestep, ensuring deterministic, predictable trajectories that keep the simulation tractable.

Figure 1 illustrates the environment. Bees are represented as green circles, unharvested flowers as purple squares, and harvested flowers as orange squares.

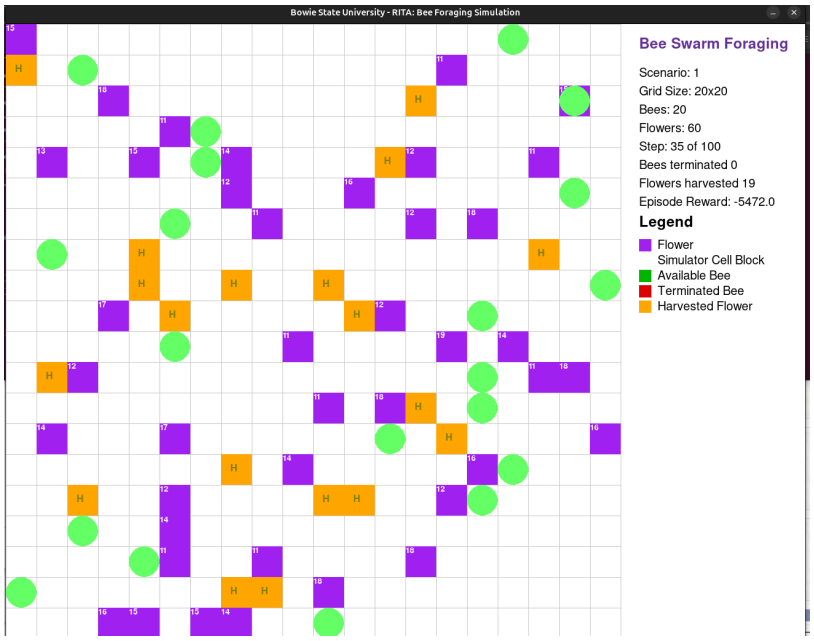

Fig. 1. Simulated bee-foraging environment. Green circles: active bee agents; purple squares: unharvested flowers; orange squares: harvested flowers.

3.2 System Assumptions

We adopt the following operating model throughout this paper. Agents are deployed on a shared 2D grid environment and execute policies entirely from local observations: each agent b_i observes only its own state $\mathbf{x}_i^{\mathrm{b}}$ and its currently assigned task set $\mathcal{T}_i$, with no access to other agents' states or actions. No inter-agent message passing occurs at any point during execution; the system is therefore immune to network delay, packet loss, and synchronization failures. A *hard deadline* on flower f_j means that pollen yield p_j is received only if the harvest time t falls within the range, $t^j_{\min} \leq t \leq t^j_{\max}$; harvesting outside this window yields $p_j = 0$ and reward $r = 0$, making the deadline a hard constraint rather than a soft one. This is distinct from a *soft deadline*, which carries a scaled penalty but does not eliminate yield entirely.

$$o_t^i = \left(\mathbf{x}_i^{\mathrm{bee}},\ \{\mathbf{x}_j^{\mathrm{flower}}\}_{j\in\mathcal{T}_i}\right) \tag{1}$$

3.3 Problem Formulation

The optimization objective is threefold: (i) maximize total pollen harvested across all agents, (ii) prioritize high-priority flowers, and (iii) minimize agent failure. Formally:

$$\max_{a\in\mathcal{A}} \left(\sum_{i=1}^{N} P_i + \lambda \sum_{j\in F_h} \rho_j - \mu \sum_{k=1}^{N} \mathbb{1}_{\mathbb{F}_k} \right) \tag{2}$$

where P_i is the total pollen collected by agent i, $F_h \subseteq F$ denotes the set of high-priority flowers, ρ_j is the priority weight of flower j, $\mathbb{1}_{\mathbb{F}_k}$ indicates whether agent k has failed, and $\lambda, \mu > 0$ are weighting coefficients balancing priorities and failure penalties. Agent failure occurs through two mechanisms:

$$\mathbb{F}_k = \begin{cases} 1 & \text{if } E_k \leq E_{\min} \text{ or } R_k = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where E_k is the current energy level of agent k, $E_{\min}$ is the minimum viable energy threshold, and $R_k \sim \text{Bernoulli}(p_{\text{fail}})$ is an independent stochastic failure indicator.

Environment Dynamics and Agent Constraints The simulation operates under the following rules:

1. **Movement.** Each agent b_i advances unidirectionally by exactly one cell per timestep; lateral, vertical, and backward movements are prohibited.
2. **Static flowers.** Flowers occupy fixed positions throughout the episode.
3. **Flower attributes.** Each flower f_j is characterized by a priority level $\rho_j \in \{0.3, 0.6, 0.99\}$ (Low/Medium/High) and a pollen yield $p_j \in \mathbb{R}^+$, both assigned at initialization.
4. **Harvesting range.** Agent b_i can harvest f_j only if $d(b_i, f_j) \leq 1$ cell (Manhattan distance).
5. **Vertical harvesting.** Agents may harvest flowers on the same row or on immediately adjacent rows (above or below), subject to grid boundaries.
6. **Task allocation.** Each agent is pre-assigned a task set $\mathcal{T}_i \subseteq F$; orphaned tasks (from failed agents) may be dynamically adopted.
7. **Grid wraparound.** Upon reaching the terminal column before timestep T, an agent returns to the first column of its current row.

8. **Energy model.** Energy consumption is: $E_{\text{move}} = 1$ unit per timestep, $E_{\text{groom}} = 5$ units per grooming action, $E_{\text{harvest}} = 10$ units per flower.
9. **Grooming.** Agents perform grooming to offload harvested pollen or recharge energy.
10. **Failure.** Agent b_i terminates when $E_i \leq E_{\min}$ or a pre-sampled stochastic failure threshold is reached.

3.4 Problem Constraints

Two primary constraints govern the optimization. First, all harvesting must occur within the time horizon $T_{\max}$. Second, each agent must maintain energy above $E_{\min}$ throughout the episode:

$$E_i(t) > E_{\min}, \quad \forall\, t \in [0, T_{\max}],\ b_i \in B \tag{4}$$

Failure conditions are formalized as:

$$E_k(t) \leq E_{\min} \quad \text{(energy depletion)} \tag{5}$$

$$R_k(t) = 1, \quad R_k(t) \sim \text{Bernoulli}(p_{\text{fail}}) \quad \text{(stochastic failure)} \tag{6}$$

where $p_{\text{fail}} = 0.3$ is the per-timestep stochastic failure probability for agents assigned a failure event. Once agent b_k fails, it is removed from the active set B_{active}, and its assigned tasks $\mathcal{T}_k$ become orphaned, triggering the retasking protocol described in Section 4.8.

3.5 Reward Function

The reward signal encodes four behavioral objectives: harvesting efficiency, maintenance, deadline satisfaction, and cooperative allocation. We design a composite, time-discounted reward as follows.

Time-Discounted Scaling All base reward components are multiplied by a time-decay factor that increases urgency as the episode progresses:

$$r(a_t) = \frac{T_{\max} - t}{T_{\max}} \cdot \bar{r}(a_t) \tag{7}$$

where $\bar{r}(a_t)$ is the base reward for action a_t and t is the current timestep. This implements reward shaping [19]: an early harvest yields higher reward than an identical late harvest, incentivizing proactive planning rather than greedy end-of-episode behavior.

Base Action Rewards Table 1 summarizes base reward values. The symmetric penalty structure (-10 for missed harvests and missed maintenance) ensures that the agent treats energy management and task execution with equal urgency, preventing degenerate policies that sacrifice maintenance for short-term reward.

Priority- and Cooperation-Aware Reward A supplementary priority reward encourages cooperative task allocation: agents should prefer lower-priority flowers when higher-priority ones can be covered by teammates. Let $\mathcal{P} = \{\text{Low}, \text{Medium}, \text{High}\}$, encoded as $\rho : \mathcal{P} \to \{0.3, 0.6, 0.99\}$. For agent b_i harvesting flower f_j:

$$r_{\text{priority}}(b_i, f_j) = \phi(b_i, f_j) \cdot \rho_j \tag{8}$$

Table 1. Base reward values $\bar{r}(a_t)$ before time-decay scaling.

Action	$\bar{r}$	Rationale
Successful harvest	+10	Primary task objective
Grooming (offload or recharge)	+10	Maintenance enabling future harvests
Do nothing	0	Neutral; no progress
Missed harvest opportunity	−10	Agent in range but did not harvest
Missed grooming (offload)	−10	Pollen capacity exceeded without offload
Missed grooming (recharge)	−10	Energy critically low without recharge
Hard-deadline harvest	+20	Bonus for satisfying hard constraints

where $\phi(b_i, f_j)$ is a cooperation coefficient:

$$\phi(b_i, f_j) = \begin{cases} R_{\text{low}} = +5 & \text{if other agents target } f_j \text{ and } \rho_j \leq \min_{f' \in \mathcal{A}_{b_i}} \rho_{f'} \\ R_{\text{alt}} = +5 & \text{if alone and lower-priority alternatives exist} \\ R_{\text{same}} = 10 & \text{if alone and all alternatives share the same priority} \\ R_{\text{none}} = +15 & \text{if alone and no alternatives remain} \end{cases} \tag{9}$$

Here $\mathcal{A}_{b_i} = \{f_j \in F \mid |row(b_i) - row(f_j)| \leq 1\}$ denotes the set of flowers reachable by b_i within ± 1 row of its current trajectory, and the scalar constants take values $R_{\text{low}} = 5$, $R_{\text{alt}} = 5$, $R_{\text{same}} = 10$, $R_{\text{none}} = 15$. This incentivizes agents to harvest any available flower when no teammates are nearby, regardless of priority, maximizing collective pollen yield.

Hard-Deadline Bonus Flowers with hard time windows receive an additional bonus when harvested within their prescribed window $[t^j_{\min}, t^j_{\max}]$:

$$r_{\text{deadline}}(b_i, f_j) = \begin{cases} 20 & \text{if } w_j = 1 \text{ and } t^j_{\min} \leq t \leq t^j_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $w_j \in \{0, 1\}$ is the hard-deadline indicator. This additive bonus makes hard-deadline flowers the highest-value targets in the reward landscape.

Total Instantaneous Reward The total reward for agent b_i at timestep t is:

$$r^i_t = \frac{T_{\max} - t}{T_{\max}} \cdot [\bar{r}(a_t) + r_{\text{deadline}}(b_i, f_j)] + r_{\text{priority}}(b_i, f_j) \tag{11}$$

The priority reward is deliberately excluded from the time-scaling factor so that cooperative behavior is incentivized uniformly throughout the episode. This reward formulation is used identically by both the DDQN and PG agents.

4 Methodology

4.1 Action Space

Each agent selects from $|\mathcal{A}_i| = 3 + |\mathcal{T}_i|$ actions: three fixed actions (DoNothing, GroomOffload, GroomRecharge) plus one harvest action per assigned flower (Table 2).

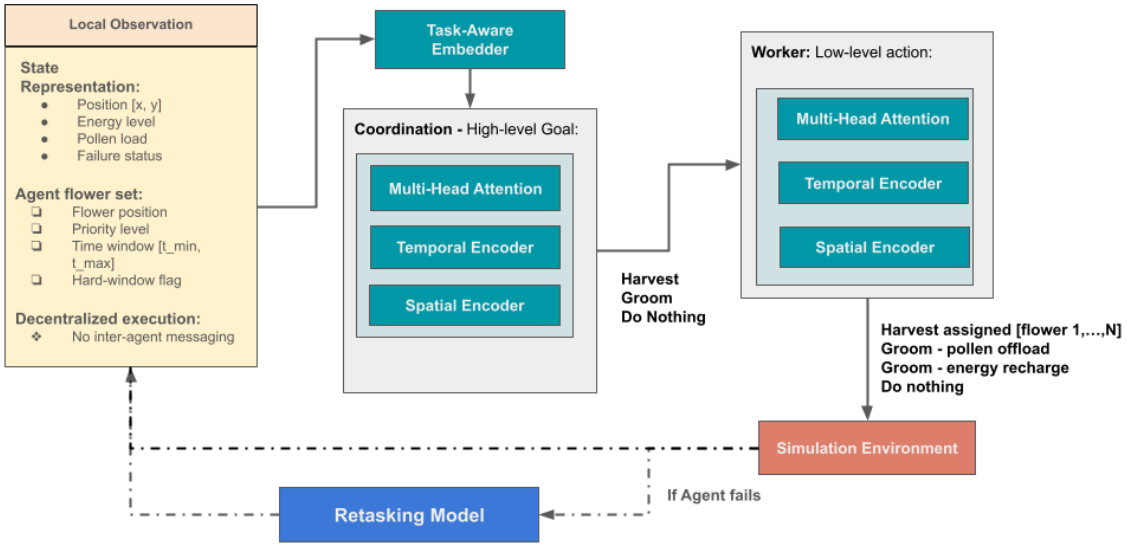

Fig. 2. Overview of the HDRL framework. Local observations are processed by the shared spatiotemporal encoder stack (Spatial, Temporal, and Task-Aware Embedder) to produce embedding $\mathbf{z}_i$. The Coordinator selects a high-level goal $g_t \in \mathcal{G}$; the Worker maps (s_t, g_t) to a primitive action $a_t \in \mathcal{A}_{\text{valid}}(g_t)$. Upon agent failure, the Neural Retasking module (dashed path) redistributes orphaned tasks to surviving agents without centralized coordination.

Table 2. Action space for bee agents.

Action	Description
Do Nothing	Agent is idle for one timestep; 1 energy unit consumed
Groom Offload	Offload collected pollen; 5 energy units consumed
Groom Recharge	Restore energy to maximum; 5 energy units consumed
Harvest f_j	Harvest assigned flower $f_j \in \mathcal{T}_i$ within distance $d \leq 1$; 10 energy units consumed

4.2 Hierarchical Coordination Framework

Both architectures follow a two-level hierarchy. Figure 2 provides an overview of the full framework. At each timestep t, the *coordinator* selects a high-level goal $g_t \in \mathcal{G}$:

$$\mathcal{G} = \{\textsc{HarvestAssigned}, \textsc{GroomOffload}, \textsc{GroomRecharge}, \textsc{DoNothing}\} \quad (12)$$

The *worker* then selects a primitive action $a_t \in \mathcal{A}$ conditioned on both s_t and g_t.

4.3 State Representation and Task-Aware Embedder

The state for each agent b_i comprises bee features $\mathbf{x}_i^{\text{bee}} \in \mathbb{R}^5$ (normalized position, energy, pollen load, and failure status) and a variable-length task set $\mathcal{T}_i$ of flower features $\mathbf{x}_j^{\text{flower}} \in \mathbb{R}^6$ (position, harvest status, time window, priority). Table 3 provides a complete breakdown of the agent state and task representations.

The task-aware embedder produces $\mathbf{z}_i \in \mathbb{R}^{64}$ by combining a bee embedding $\mathbf{h}_{\text{bee}} = \text{MLP}_{\text{bee}}(\mathbf{x}_i^{\text{bee}}) + \mathbf{s}^i_{\text{spatial}} + \mathbf{t}_{\text{embed}}$ with a task summary $\mathbf{h}_{\text{tasks}} = \text{Attention}(\mathbf{q}, \{\text{MLP}_{\text{task}}([\mathbf{x}_j^{\text{flower}}; u_j; w_j]) + \mathbf{s}^j_{\text{spatial}}\})$, where $\mathbf{q} \in \mathbb{R}^{32}$ is a learnable query. The attention mechanism yields a permutation-invariant, fixed-dimensional embedding regardless of task-set cardinality.

Table 3. State representation for agents and flowers.

Feature	Dim	Range	Description
Bee state ($\mathbf{x}_i^{\text{bee}} \in \mathbb{R}^5$)			
Position	2	$[0,1]^2$	Normalized (x, y)
Pollen load	1	$[0,1]$	Collected / capacity
Terminated	1	$\{0,1\}$	Failure indicator
Energy	1	$[0,1]$	Current / maximum
Flower state ($\mathbf{x}_j^{\text{flower}} \in \mathbb{R}^6$)			
Position	2	$[0,1]^2$	Normalized (x, y)
Harvested	1	$\{0,1\}$	Completion flag
Time window	2	$[0,1]^2$	$(t_{\min}, t_{\max})/T$
Priority	1	$\{0.3, 0.6, 0.99\}$	Low / Medium / High

4.4 Spatial Encoder

A key architectural innovation is the spatial encoder, which enables explicit distance-aware reasoning. It computes two complementary representations.

Sinusoidal position encoding. Absolute locations are encoded as:

$$\text{PE}(\mathbf{p}, 2k) = \sin\left(\frac{\mathbf{p} \cdot \omega_k}{g}\right), \quad \text{PE}(\mathbf{p}, 2k+1) = \cos\left(\frac{\mathbf{p} \cdot \omega_k}{g}\right) \tag{13}$$

where $\mathbf{p} = (x, y)$, g is the grid size, $\omega_k = 2^k$, and $k \in \{0, \ldots, d/8 - 1\}$ spans multiple spatial scales.

Distance-based features. Five statistics per entity i capture pairwise spatial relationships: minimum, mean, and maximum pairwise distances ($d^i_{\min}$, d^i_{mean}, $d^i_{\max}$), their standard deviation σ^i_d, and r_i, the fraction of entities within harvesting range.

Gated combination. Position embeddings $\mathbf{h}_{\text{pos}}$ and distance embeddings $\mathbf{h}_{\text{dist}}$ are fused via a learned sigmoid gate α: $\mathbf{s}_{\text{spatial}} = \alpha \odot \mathbf{h}_{\text{pos}} + (1 - \alpha) \odot \mathbf{h}_{\text{dist}}$, adaptively weighting positional versus relational information. A binary harvestability flag $h_j = \mathbb{1}[d(b_i, f_j) \leq d_{\text{harvest}}]$ prevents invalid harvest actions.

4.5 Temporal Encoder

The temporal encoder enables deadline-aware reasoning via sinusoidal embeddings at logarithmically-spaced frequencies $\nu_k = \exp(-k \log(T_{\max})/(d/2))$, processed through a temporal MLP to yield $\mathbf{t}_{\text{embed}}$. Per-flower urgency $u_j = 1 - \max(0, t^j_{\max} - t)/T_{\max} \in [0, 1]$ rises toward 1 as a deadline approaches; a binary flag $w_j = \mathbb{1}[\text{hard deadline}]$ distinguishes hard from soft constraints.

4.6 Double Deep Q-Network Architecture

Standard Q-learning suffers from systematic maximization bias: using a single network for both action selection and value evaluation causes overestimated Q-values, destabilizing training [10]. Our hierarchical agent corrects this via DDQN [27], maintaining online parameters ϕ and frozen target parameters ϕ'.

Table 4. Hyperparameters for DDQN and Policy Gradient (PG).

Hyperparameter	**DDQN**	**PG**
Discount factor γ	0.99	0.99
Learning rate	3×10^{-5} (worker) 1×10^{-3} (goal)	0.005
Replay buffer capacity	50,000	—
Minibatch size B	64	—
ϵ schedule	1.0→0.05 (linear)	—
Target update frequency	100 steps	—
Update interval	—	10 steps
Value loss coeff. α	—	0.5
Entropy bonus coeff. β	—	0.01
Gradient clip norm	1.0	1.0
Embedding dimension d	256	64
Attention heads	8	4

Goal selector (coordinator). The coordinator maps the state embedding $\mathbf{z}_t$ and a resource context vector $\mathbf{c}_t = [E_t/E_{\max}, L_t/L_{\max}, t/T_{\max}]$ to goal Q-values $Q_\theta(s_t, g) = \mathrm{MLP}_{\mathrm{goal}}(\mathrm{Fusion}([\mathbf{z}_t; \mathrm{MLP}_{\mathrm{ctx}}(\mathbf{c}_t)]))$.

Worker network. The worker projects the goal into embedding $\mathbf{g}_{\mathrm{emb}} \in \mathbb{R}^{d/2}$ and refines representations through a three-stage multi-head attention cascade: bee-self attention ($\tilde{\mathbf{B}}$), bee-flower cross-attention ($\mathbf{C}_t$), and flower-self attention ($\tilde{\mathbf{F}}$). The attended features are concatenated and pooled to form per-flower feature vectors $\mathbf{f}_j = [\bar{\mathbf{v}}_t; \bar{\mathbf{C}}_t; \tilde{\mathbf{F}}_j; \mathbf{g}_{\mathrm{emb}}]$, from which base Q-values $Q_\phi^{\mathrm{base}}(s_t, a_j \mid g_t) = \mathrm{MLP}_Q(\mathbf{f}_j)$ are computed.

Distance-aware Q-value shaping. Because the harvest reward is temporally delayed, Q-values are augmented with $\Delta Q_j = \mathbb{1}[d_j \le d_{\mathrm{harvest}}] + \tanh(\mathrm{MLP}_{\mathrm{dist}}(d_j)) \cdot \lambda_{\mathrm{dist}}$, where λ_{dist} is a learnable scalar. The hard indicator provides an immediate bonus for in-range flowers; the tanh term encodes a smooth distance-to-value gradient that accelerates exploration without altering the optimal policy [19].

Goal-conditioned action masking. Invalid actions are suppressed via a goal-conditioned mask:

$$\mathcal{A}_{\mathrm{valid}}(g_t) = \begin{cases} \{\mathrm{harvest}_j\}_{j=1}^{|\mathcal{T}_i|} & g_t = \textsc{Harvest} \\ \{\textsc{GroomOffload}, \textsc{GroomRecharge}\} & g_t \in \{\textsc{Groom-*}\} \\ \{\textsc{DoNothing}\} & g_t = \textsc{DoNothing} \end{cases} \tag{14}$$

Action selection follows an ϵ-greedy policy restricted to $\mathcal{A}_{\mathrm{valid}}(g_t)$, with ϵ_t linearly from 1.0 to 0.05 over the full training horizon.

DDQN update rule. A minibatch of size $B = 64$ is sampled from replay buffer $\mathcal{D}$. The Double DQN target decouples action selection (online network) from value evaluation (frozen target): $y_t = r_t + \gamma(1-d_t)\, Q_{\phi'}(s_{t+1}, \arg\max_{a'} Q_\phi(s_{t+1}, a' \mid g_{t+1}) \mid g_{t+1})$, eliminating maximization bias [10, 27]. The loss $\mathcal{L}(\phi) = \mathbb{E}[(Q_\phi(s, a \mid g) - y)^2]$ is minimized with gradient clipping. Target parameters ϕ' are synchronized via hard copy. Hyperparameters are listed in Table 4.

4.7 Policy Gradient Architecture

The policy gradient agent shares the same hierarchical coordinator–worker decomposition and spatiotemporal encoder stack as the DDQN agent, but replaces value-based Q-learning with a stochastic policy trained via the REINFORCE algorithm [32].

Coordinator: goal distribution. The *coordinator* maps $\mathbf{z}_t$ to $\pi_\psi(g_t \mid s_t) = \text{Softmax}(\text{MLP}_{\text{goal}}(\mathbf{z}_t) \odot (1 - \mathbf{m}_t))$, where resource-aware mask $\mathbf{m}_t$ suppresses semantically invalid goals (e.g., GROOMOFFLOAD when $L_t < 1$). A value head $V_\psi(s_t) = \text{MLP}_{\text{value}}(\mathbf{z}_t)$ provides the advantage baseline.

Worker: action distribution. The worker produces $\pi_\omega(a_t \mid s_t, g_t) = \text{Softmax}(\boldsymbol{\ell}_t \cdot \mathbb{1}_{\mathcal{A}_{\text{valid}}(g_t)})$ by masking logits to the goal-valid action set.

REINFORCE update with baseline. The discounted return from timestep t is $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$. Baseline-subtracted advantage estimates:

$$\hat{A}_t = R_t - V_\psi(s_t) \tag{15}$$

The joint policy gradient loss for both levels of the hierarchy is:

$$\begin{aligned} \mathcal{L}_{\text{PG}}(\psi, \omega) = -\mathbb{E}_\tau \left[\sum_{t=0}^{T} \hat{A}_t \big(\log \pi_\psi(g_t \mid s_t) + \log \pi_\omega(a_t \mid s_t, g_t) \big) \right] \\ + \alpha \, \mathcal{L}_{\text{value}}(\psi) - \beta \, \mathcal{H}(\pi_\omega) \end{aligned} \tag{16}$$

where $\mathcal{L}_{\text{value}}(\psi) = \mathbb{E}[(R_t - V_\psi(s_t))^2]$, $\mathcal{H}(\pi_\omega) = -\mathbb{E}[\log \pi_\omega(a_t \mid s_t, g_t)]$ is a policy entropy bonus discouraging premature convergence, and α, β are scalar coefficients [17].

4.8 Neural Retasking Mechanism

When agent b_k fails at time t_{fail}, its assigned tasks $\mathcal{T}_k$ become orphaned. For each orphaned task $f_o \in \mathcal{T}_k$ and candidate agent $b_i \in B_{\text{active}}$, the retasking model computes an adoption probability $p_{\text{adopt}}(b_i, f_o)$ through four specialized encoders.

Bee and flower encoders. Candidate bees and orphaned flowers are encoded as $\mathbf{h}_i^{\text{bee}} = \text{MLP}_{\text{bee}}(\mathbf{x}_i^{\text{bee}})$ and $\mathbf{h}_o^{\text{flower}} = \text{MLP}_{\text{flower}}(\mathbf{x}_o^{\text{flower}})$.

Path efficiency encoder. Three geometric features quantify whether the orphaned flower lies near the agent's trajectory: normalized distance d_{norm}, angular deviation θ_{dev}, and path efficiency $s_{\text{path}} = \exp(-\Delta d/5)$, where Δd is the detour cost. These are encoded as $\mathbf{h}_{\text{path}} = \text{MLP}_{\text{path}}([d_{\text{norm}}; \theta_{\text{dev}}; s_{\text{path}}])$.

Task context encoder. A workload score $w_i = |\mathcal{T}_i| + 0.5 n_{\text{urgent}} + 0.3 n_{\text{high-priority}}$ summarizes the agent's current burden, and is encoded alongside average urgency $\bar{u}_i$ and remaining capacity c_i as $\mathbf{h}_{\text{context}} = \text{MLP}_{\text{context}}([|\mathcal{T}_i|; \bar{u}_i; w_i; c_i])$.

Decision network. The four encodings are aggregated via 4-head MHA and decoded to adoption probability $\mathbf{p}_{\text{adopt}} = \text{Softmax}(\text{MLP}_{\text{decision}}(\text{MHA}([\mathbf{h}_i^{\text{bee}}; \mathbf{h}_o^{\text{flower}}; \mathbf{h}_{\text{path}}; \mathbf{h}_{\text{context}}]))) \in \mathbb{R}^2$. The adoption decision is adopt $= \mathbb{1}[p_{\text{adopt}}[1] > 0.5]$.

Training objective. The retasking model is trained via cross-entropy loss with ℓ_2 regularization:

$$\mathcal{L}_{\text{retask}} = -\mathbb{E}_{(b_i,f_o,y)}[y \log p_{\text{adopt}}[1] + (1-y)\log p_{\text{adopt}}[0]] + \lambda\|\theta\|_2^2 \tag{17}$$

where $y \in \{0,1\}$ indicates whether reassignment improved system performance. For the PG agent, the retasking log-probability is incorporated into the overall policy gradient loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PG}}(\psi,\omega) - \log p_{\text{adopt}} \tag{18}$$

This end-to-end gradient flow trains the retasking model jointly with the main policy, incentivizing adoption decisions consistent with the agent's global return objective.

4.9 MILP Baseline

We implement MILP [5] as an offline optimum baseline. To align the MILP formulation with the RL environment, grid wraparound is replaced by replicated identical grids, allowing bees to transition to the first column of a new grid replica when a wraparound is needed. Time is discretized by column index: a bee advancing from column 2 to column 8 has consumed 6 timesteps.

Two binary variable families are used: $h(b_i, i, j)$ for same-column harvesting (of flowers directly above, below, or at the bee's current position) and $ah(b_i, i, j)$ for advance harvesting (the column immediately ahead). The MILP enforces five constraint families: (1) *inability to harvest* — flowers absent from adjacent rows and the first column cannot be advance-harvested; (2) *single harvest per bee* — each bee harvests at most one flower per timestep; (3) *single harvest per flower* — each flower is harvested by at most one bee; (4) *time-window constraints* — harvest variables are set to zero outside a flower's prescribed window; (5) *grid consistency* — harvested flowers remain harvested across all grid replicas. The MILP provides a deterministic upper bound on pollen yield under perfect information and no agent failures, serving as an idealized performance reference.

5 Experiments and Results

5.1 Experimental Setup

We evaluate DDQN and PG against the MILP baseline across a systematic sweep of temporal horizons. All experiments are governed by a single YAML configuration file that controls model paths, environment overrides, the timestep sweep, the number of independent runs, and figure generation, enabling any subset of models to be evaluated in isolation without code modification.

Environment Configuration The simulation uses a 12×12 grid with $N = 10$ bee agents and $M = 100$ flowers per scenario. Flowers are assigned priority levels (Low: 0.3, Medium: 0.6, High: 0.99) and time windows $[t_{\min}, t_{\max}]$ designating valid harvesting periods; a hard-deadline flag $w_j \in \{0,1\}$ marks the most time-critical subset. Agent dynamics incorporate three resource constraints: energy ($E_{\max} = 2{,}000$ units), pollen storage capacity ($C_{\max} = 100$ units), and stochastic failures in which 30% of agents are assigned a failure timestep at scenario initialization. Full configuration details are provided in Table 5.

Table 5. Training and evaluation configuration.

Category	Parameter	Value
Environment	Grid size	12×12
	Number of bees	10
	Number of flowers	100
	Training episodes	300
	Max energy	2,000
	Max pollen capacity	100
	Stochastic failure rate	30%
Evaluation	Timestep range	100–1000 (step 100)
	Scenarios per cell	10
	Independent runs per cell	3

Table 6. Pollen yield, total flowers harvested, and hard-deadline completion rate (HD%) for DDQN, PG against the MILP baseline at representative temporal horizons $T \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$. Values are mean $\pm$ std over evaluation scenarios. **Bold**: best DRL result per column group. †: model exceeds the MILP upper bound.

T	Pollen harvested (mean $\pm$ std)			Flowers harv.		HD%	
	DDQN	PG	MILP	DDQN	PG	DDQN	PG
100	$\mathbf{783.0 \pm 26.9}$	657.0 ± 43.8	1011.0	**52**	42	**46.6**	26.7
200	749.0 ± 53.7	$\mathbf{964.0 \pm 137.2}$	1098.5	50	**63**	**40.4**	**40.4**
300	771.0 ± 91.9	$\mathbf{983.5 \pm 31.8}$	1100.5	50	**64**	**44.3**	39.6
400	671.0 ± 77.8	$\mathbf{1029.0 \pm 17.0}^{\dagger}$	971.0	44	**68**	44.1	**47.7**
500	752.5 ± 55.9	$\mathbf{1024.5 \pm 0.7}^{\dagger}$	1010.5	51	**68**	45.6	**50.5**
600	738.0 ± 17.0	$\mathbf{1153.5 \pm 16.3}^{\dagger}$	1045.0	50	**77**	45.0	**59.5**
700	783.0 ± 14.1	$\mathbf{1223.0 \pm 63.6}^{\dagger}$	1071.0	52	**81**	47.2	**66.7**
800	766.0 ± 2.8	$\mathbf{1264.5 \pm 98.3}^{\dagger}$	1073.5	50	**83**	42.2	**62.2**
900	715.0 ± 18.4	$\mathbf{1193.0 \pm 66.5}^{\dagger}$	1045.0	48	**80**	41.2	**65.8**
1000	733.5 ± 12.0	$\mathbf{1159.5 \pm 10.6}^{\dagger}$	1004.0	49	**78**	43.4	**61.9**

Evaluation Protocol For each model $m \in \{\text{DDQN}, \text{PG}\}$ and horizon $T \in \{100, 200, \ldots, 1000\}$, we execute $R = 3$ independent runs of $S = 10$ held-out scenarios, reporting mean $\pm$ std over the resulting 30 instances per cell.

The MILP baseline (Section 4.9) provides an idealized upper bound: which is a pre-computed assignments evaluated on the same held-out scenarios and all flowers had a hard deadline time-window.

5.2 Results

Statistical significance. All pairwise comparisons between DDQN and PG at each temporal horizon are evaluated using two-sided Wilcoxon rank-sum tests ($\alpha = 0.05$) over the 30 evaluation instances per cell ($R = 3$ runs $\times$ $S = 10$ scenarios). DDQN's advantage at $T = 100$ is statistically significant for pollen yield ($p < 0.01$), flowers harvested ($p < 0.01$), and HD% ($p < 0.01$). PG's advantage in pollen yield and flowers harvested is statistically significant for all $T \geq 200$ ($p < 0.05$). The HD% crossover reaches significance at $T \geq 500$ ($p < 0.05$). PG's super-optimal pollen yield relative to MILP is statistically significant for all $T \geq 400$ ($p < 0.01$), confirming that learned failure recovery, rather than evaluation variance, drives the performance gap.

Table 7. Harvest efficiency (% of total available pollen), performance relative to the MILP baseline (% of MILP objective), and hard-deadline completion rate (HD%) across all evaluated horizons for DDQN, PG. **Bold**: best DRL result per column group. †: model exceeds the MILP upper bound.

	Harvest eff. (%)		**% of MILP**		**HD%**	
T	**DDQN**	**PG**	**DDQN**	**PG**	**DDQN**	**PG**
100	**51.8**	43.5	**77.4**	65.0	**46.6**	26.7
200	49.1	**63.2**	68.2	**87.8**	**40.4**	**40.4**
300	50.0	**63.7**	70.1	**89.4**	**44.3**	39.6
400	44.7	**68.5**	69.1	**106.0**†	44.1	**47.7**
500	50.8	**69.2**	74.5	**101.4**†	45.6	**50.5**
600	49.6	**77.5**	70.6	**110.4**†	45.0	**59.5**
700	52.1	**81.5**	73.1	**114.2**†	47.2	**66.7**
800	50.5	**83.4**	71.4	**117.8**†	42.2	**62.2**
900	47.6	**79.4**	68.4	**114.2**†	41.2	**65.8**
1000	49.3	**78.0**	73.1	**115.5**†	43.4	**61.9**

Table 8. Flowers harvested by priority level (High: $\rho = 0.99$; Medium: $\rho = 0.60$; Low: $\rho = 0.30$) for DDQN, PG. Each cell shows *harvested / available* flowers per scenario. **Bold**: best DRL result within each priority group.

T	**High** ($\rho = 0.99$)		**Medium** ($\rho = 0.60$)		**Low** ($\rho = 0.30$)	
	DDQN	**PG**	**DDQN**	**PG**	**DDQN**	**PG**
100	17.0 / 32	**18.0** / 32	**25.0** / 40	11.0 / 40	**13.0** / 28	12.0 / 28
200	16.5 / 40	**23.5** / 40	17.0 / 34	**19.0** / 34	14.0 / 26	**21.0** / 26
300	18.5 / 34	**24.0** / 34	21.0 / 40	**25.0** / 40	13.0 / 26	**17.0** / 26
400	18.0 / 37	**27.0** / 37	13.0 / 27	**18.0** / 27	16.0 / 36	**26.0** / 36
500	11.5 / 23	**18.5** / 23	27.0 / 42	**30.0** / 42	14.0 / 35	**23.0** / 35
600	13.0 / 33	**25.5** / 33	21.0 / 36	**28.0** / 36	**19.0** / 31	**19.0** / 31
700	17.0 / 34	**27.0** / 34	20.0 / 39	**33.0** / 39	11.0 / 27	**21.0** / 27
800	17.5 / 40	**31.0** / 40	16.0 / 34	**27.0** / 34	14.0 / 26	**24.0** / 26
900	15.5 / 32	**26.5** / 32	18.0 / 36	**32.0** / 36	19.0 / 32	**24.0** / 32
1000	16.5 / 33	**26.5** / 33	18.0 / 35	**26.0** / 35	15.0 / 32	**21.0** / 32

Tables 6 and 8 and Figure 3 summarize results averaged over 30 evaluation instances per timestep cell (3 runs $\times$ 10 scenarios). Figure 3 evaluates total pollen yield under a maximally constrained scenario in which all 100 flowers carry hard time-window constraints—the same assumption embedded in the MILP formulation—providing a tight, constraint-matched upper bound for comparison.

Pollen yield and hard-deadline satisfaction. The two DRL agents diverge sharply after $T = 100$ (Table 6, Figure 3). At $T = 100$, DDQN leads across every metric: 783 ± 27 pollen units (77.4% of MILP) vs. PG's 657 ± 44 (65.0%), 52 vs. 42 flowers harvested, and HD% of 46.6% vs. 26.7%. DDQN's replay buffer resamples rare deadline-critical transitions that on-policy REINFORCE encounters only once per rollout, explaining this short-horizon advantage. From $T = 200$ onward PG surpasses DDQN without exception. By $T = 300$, PG yields 984 ± 32 units (89.4% of MILP) while DDQN yields 771 ± 92 (70.1%). PG first exceeds the MILP upper bound at $T = 400$ (106.0%) and peaks at $T = 800$ with 117.8% of the MILP objective, reaching HD% of 66.7% at $T = 700$ against DDQN's 47.2%. DDQN plateaus at 68–77% of MILP and does not exceed it at any horizon. The super-optimal PG

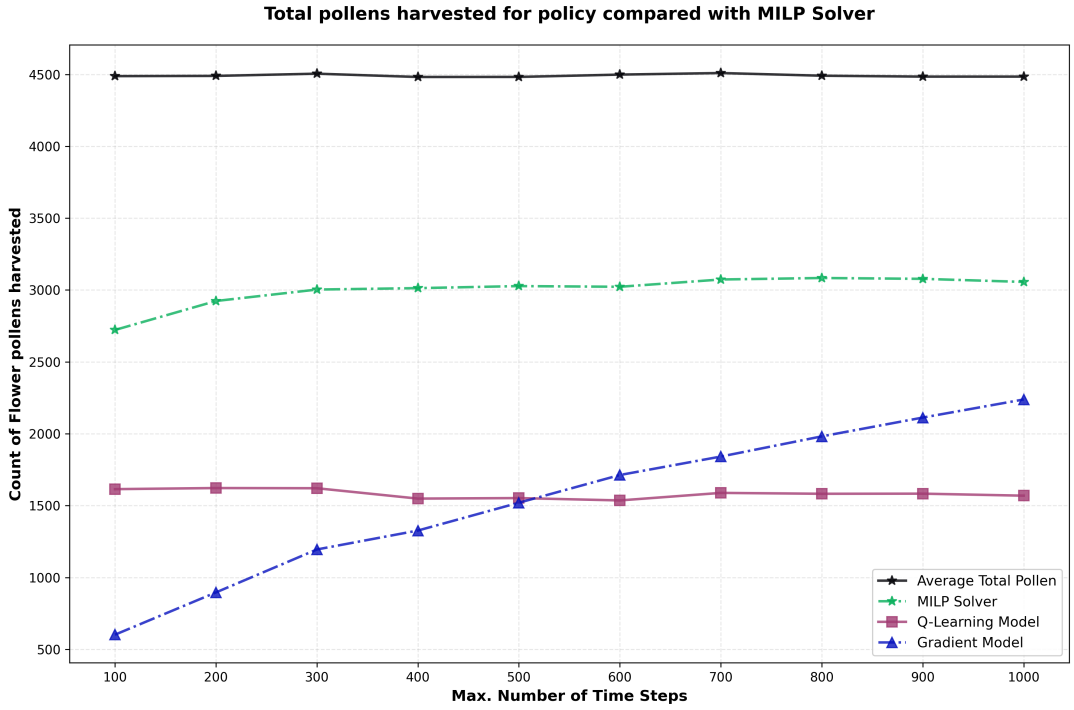

Fig. 3. Total pollen harvested (y-axis: cumulative pollen units across all $N = 10$ agents per episode) versus maximum timestep budget T (x-axis, 100–1000) in the all-hard-time-window scenario, in which every flower carries a hard deadline constraint ($w_j = 1$) and any harvest outside $[t^j_{\min}, t^j_{\max}]$ yields zero pollen. DDQN (blue), Policy Gradient (orange), and the MILP baseline (green dashed) are compared; the black reference line indicates mean total pollen available per scenario. The MILP operates under perfect information with no agent failures, providing a constraint-matched upper bound (≈970–1,100 units depending on horizon). Values are means over 30 evaluation instances (3 runs × 10 scenarios); shaded bands indicate ±1 standard deviation. PG surpasses the MILP bound from $T = 400$ onward via learned failure recovery; DDQN leads only at $T = 100$.

yields arise because the learned retasking mechanism recovers yield from the 30% of agents that fail mid-episode—failures the static MILP cannot anticipate.

Flower count and priority coverage. DDQN's flower count is largely flat, ranging from 44 to 52 across all horizons (with a dip to 44 at $T = 400$), indicating convergence to a spatially fixed harvest strategy. PG grows progressively: 64 flowers at $T = 300$, 81 at $T = 700$, and 78 at $T = 1000$. Priority breakdown (Table 8) confirms broader coverage by PG: at $T = 700$, PG achieves 79.4% (High), 84.6% (Medium), and 77.8% (Low) completion rates, compared with DDQN's 50.0%, 51.3%, and 40.7%. PG's stochastic policy with entropy regularization encourages coverage across all tiers; DDQN's greedy selection concentrates on highest-immediate-reward flowers and systematically under-harvests lower-priority tasks. The sole DDQN exception—leading on Medium-priority at $T = 100$ (25.0 vs. 11.0 of 40)—reflects replay-buffer overrepresentation of early-episode encounters near agent starting positions.

6 Discussion

The results in Tables 6–8 and Figure 3 surface five key insights with implications for the design of multi-agent systems operating under hard temporal constraints.

Horizon length is the primary determinant of algorithm advantage. Neither DDQN nor PG uniformly dominates: superiority is horizon-dependent and the crossover is abrupt. DDQN leads at $T = 100$ on every metric—pollen yield (783 vs. 657 units), flowers harvested (52 vs. 42), and HD% (46.6% vs. 26.7%)—because its replay buffer efficiently reuses the rare deadline-critical transitions available in short episodes. From $T = 200$ onward, PG's long-horizon Monte Carlo returns integrate the full consequence of each action. In the all-hard-window setting, the crossover is sharp: PG achieves 106% of the MILP objective at $T = 400$ while DDQN reaches only 69.1%, a gap that widens to 114.2% vs. 73.1% at $T = 700$. This carries direct deployment implications: DDQN is the appropriate algorithm for tight real-time budgets ($T \leq 300$), while PG is superior whenever the episode length permits extended task redistribution.

PG's super-optimal yield is meaningful against a constraint-matched MILP. The all-hard-window evaluation in Figure 3 is the most demanding test of deadline performance, because the MILP baseline is formulated under the same constraint regime: every flower carries a hard deadline, and the MILP operates under perfect information. That PG nonetheless exceeds the MILP from $T = 400$ onward—reaching 117.8% of the MILP objective at $T = 800$—establishes that the performance gap is not an artifact of mismatched evaluation conditions. The neural retasking module responds in real time to agent failures by redistributing orphaned tasks to the nearest available agents weighted by path efficiency and workload context, recovering yield the static plan permanently abandons. DDQN's failure to achieve equivalent super-optimal behavior (peaking at 77.4% of MILP) suggests that Q-value bootstrapping introduces a temporal credit-assignment lag that limits long-horizon task redistribution under maximum deadline pressure.

Off-policy learning confers a measurable short-horizon deadline advantage. The HD% gap at $T = 100$ (46.6% vs. 26.7%, Table 6) is not explained by architectural differences—both models share the same spatiotemporal encoder, hierarchical decomposition, and retasking module. The decisive mechanism is the replay buffer, which stores and resamples deadline-critical harvest transitions that on-policy REINFORCE encounters only once per rollout. Soft-deadline flowers in the evaluation dataset further dilute the probability that on-policy rollouts naturally encounter hard-deadline transitions, while DDQN's buffer preserves and oversamples them independently of their frequency in the current policy's trajectory. This finding aligns with the RL literature suggesting that off-policy methods are better suited to sparse, deadline-constrained rewards [14].

PG achieves broader priority coverage as horizon length increases. Table 8 shows that PG's stochastic policy with entropy regularization progressively covers all three priority tiers as T grows, whereas DDQN's greedy selection concentrates on highest-immediate-reward flowers and systematically under-harvests Medium and Low priority tasks. At $T = 700$, PG's completion rates are 79.4% (High), 84.6% (Medium), and 77.8% (Low), compared with DDQN's 50.0%, 51.3%, and 40.7%—gaps of approximately 30 percentage points across all tiers. DDQN's one exception—leading on Medium-priority at $T = 100$ (25.0 vs. 11.0 of 40 available)—reflects buffer overrepresentation of early-episode Medium-priority encounters near agent starting positions, an artifact that disappears once episode length provides sufficient exploration coverage.

Hierarchical decomposition separates deadline reasoning from resource management. The coordinator–worker architecture decouples high-level goal selection—prioritize hard-deadline flowers versus perform grooming—from low-level action execution. In the all-hard-window setting this separation is especially critical: a flat policy must simultaneously learn energy management and task urgency from a single reward signal, creating competing gradient contributions that destabilize training. The goal-conditioned

action masking enforces semantic consistency between levels, ensuring that grooming goals suppress all harvest actions in the worker distribution regardless of Q-value magnitude or policy probability, and preventing contextually invalid actions throughout.

Limitations and future work.

Despite promising results, the study identifies four limitations that direct future research. First, a crossover between algorithm preferences exists: DDQN outperforms PG at $T = 100$ on pollen yield (783 ± 27 vs. 657 ± 44 units), flowers harvested (52 vs. 42), and hard-deadline completion (46.6% vs. 26.7%), yet PG surpasses DDQN on every metric from $T = 200$ onward, peaking at 117.8% of the MILP objective at $T = 800$ while DDQN plateaus at 68–77% of MILP across all horizons (Table 7); characterizing the conditions that favor each algorithm requires a systematic sweep of the hard-to-soft deadline ratio. Second, an unexplained asymmetry at short horizons shows DDQN leading on Medium-priority flowers (25.0 vs. 11.0 of 40 available, Table 8) while PG holds a marginal advantage on High-priority flowers (18.0 vs. 17.0), suggesting a deeper investigation into the relative contributions of the replay buffer, ϵ-greedy schedule, and distance-aware Q-value shaping is necessary. Third, the decentralized retasking module makes adoption decisions independently per agent without coordination among candidates, which does not guarantee globally optimal task redistribution under simultaneous failures or high task contention; lightweight auction mechanisms or learned communication protocols [7] could mitigate these inefficiencies. Finally, both agents are trained for 300 episodes on a fixed 12×12 grid, and DDQN's flat pollen yield across all horizons (733–783 units, Table 6) suggests insufficient convergence at longer time budgets; the spatial encoder's $O(N^2)$ pairwise distance computation and the retasking module's $O(|B_{\text{active}}| \times |\mathcal{T}_k|)$ per-failure cost also leave scalability uncharacterized, motivating curriculum learning and evaluation on larger grids as future work.

7 Conclusion

We have presented a hierarchical deep reinforcement learning framework with explicit spatiotemporal encoding for decentralized multi-agent task allocation under hard temporal constraints. By treating spatial distance and deadline urgency as first-class architectural components—rather than latent features to be discovered—our framework equips agents with the inductive biases necessary to make reliable decisions in time-critical, resource-constrained settings. A neural retasking mechanism further enables graceful degradation under stochastic agent failures without any centralized coordination.

Experiments on a multi-agent bee-foraging benchmark reveal a horizon-dependent crossover: DDQN substantially outperforms Policy Gradient under tight time budgets ($T \leq 300$ timesteps), achieving superior pollen yield, flower count, and hard-deadline completion, while PG surpasses DDQN from $T = 200$ onward and exceeds the MILP optimum from $T = 400$ onward—reaching 117.8% of the MILP objective at $T = 800$. These super-optimal yields confirm that the learned retasking mechanism recovers yield that static offline planning cannot anticipate. These results establish hierarchical spatiotemporal encoding as a practical and scalable approach for real-time multi-agent coordination, with direct applicability to robotics, logistics, and autonomous systems operating under strict temporal deadlines and uncertain agent availability.

Acknowledgments

This material is based on work supported by the Air Force Research Laboratory (AFRL) under Air Force Contract No. FA955023D0001. Any opinions, findings, conclusions, or

recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL.

This work was conducted in the Autonomous Technologies Lab at Bowie State University. The authors thank members of the BSU Autonomous Technologies Lab and RITA-UARC for helpful discussions.

References

1. El Alami, H., Rawat, D.B.: Reinforcement Learning-enabled Satellite Constellation Reconfiguration and Retasking for Mission-Critical Applications. arXiv preprint arXiv:2409.02270 (2024).
2. Mao, L., Ma, Z., Li, X.: A Multi-Task Dynamic Weight Optimization Framework Based on Deep Reinforcement Learning. Applied Sciences **15**(5), 2473 (2025).
3. Choudhury, S., Gupta, J.K., Morales, M.J., Kochenderfer, M.J.: Multi-Robot Task Allocation under Time-Window Constraints. In: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 32, pp. 78–86 (2022).
4. Cui, D., Peng, Z., Li, K., Li, Q., He, J., Deng, X.: A Novel Cloud Task Scheduling Framework using Hierarchical Deep Reinforcement Learning for Cloud Computing. PLoS ONE **20**(8), e0329669 (2025).
5. Floudas, C.A., Lin, X.: Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications. Annals of Operations Research **139**(1), 131–162 (2005).
6. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual Multi-Agent Policy Gradients. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018).
7. Foerster, J., Assael, I.A., De Freitas, N., Whiteson, S.: Learning to Communicate with Deep Multi-Agent Reinforcement Learning. Advances in Neural Information Processing Systems **29** (2016).
8. Gerkey, B.P., Matarić, M.J.: A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. The International Journal of Robotics Research **23**(9), 939–954 (2004).
9. Gronauer, S., Diepold, K.: Multi-Agent Deep Reinforcement Learning: A Survey. Artificial Intelligence Review **55**(2), 895–943 (2022).
10. Van Hasselt, H.: Double Q-learning. Advances in Neural Information Processing Systems **23** (2010).
11. Jia, X., Wu, P., Chen, L., Liu, H., Li, J., Yan, J.: HDGT: Heterogeneous Driving Graph Transformer for Multi-Agent Trajectory Prediction via Scene Encoding. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(11), 13860–13875 (2023).
12. Korsah, G.A., Stentz, A., Dias, M.B.: A Comprehensive Taxonomy for Multi-Robot Task Allocation. The International Journal of Robotics Research **32**(12), 1495–1512 (2013).
13. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.B.: Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. Advances in Neural Information Processing Systems **29**, 3682–3690 (2016).
14. Kumar, A., Hong, J., Singh, A., Levine, S.: Should I Run Offline Reinforcement Learning or Behavioral Cloning? In: International Conference on Learning Representations (2022).
15. Lin, L.J.: Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. Machine Learning **8**(3–4), 293–321 (1992).
16. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., et al.: Human-Level Control through Deep Reinforcement Learning. Nature **518**(7540), 529–533 (2015).
17. Mnih, V., Badia, A.P., Mirza, M., Graves, A., et al.: Asynchronous Methods for Deep Reinforcement Learning. In: International Conference on Machine Learning, pp. 1928–1937. PMLR (2016).
18. Nachum, O., Gu, S., Lee, H., Levine, S.: Data-Efficient Hierarchical Reinforcement Learning. Advances in Neural Information Processing Systems **31** (2018).
19. Ng, A.Y., Harada, D., Russell, S.: Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping. In: International Conference on Machine Learning, vol. 99, pp. 278–287 (1999).
20. Oroojlooy, A., Hajinezhad, D.: A Review of Cooperative Multi-Agent Deep Reinforcement Learning. Applied Intelligence **53**(11), 13677–13722 (2023).
21. Özbakır, L., Baykasoğlu, A., Tapkan, P.: Bees Algorithm for Generalized Assignment Problem. Applied Mathematics and Computation **215**(11), 3782–3795 (2010).
22. Rashid, T., Samvelyan, M., De Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S.: QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In: International Conference on Machine Learning, pp. 4295–4304. PMLR (2018).
23. Suslova, I., Fazli, P.: Distributed Constraint Optimization for Multi-Robot Task Allocation. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1898–1900 (2020).

24. Sutton, R.S., Precup, D., Singh, S.: Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. Artificial Intelligence **112**(1–2), 181–211 (1999).
25. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge, MA (2018).
26. Tang, G., Tang, C., Claramunt, C., Hu, X., Zhou, P.: Autonomous Mobile Robot Navigation in Unknown Environments using Deep Reinforcement Learning. Sensors **21**(19), 6422 (2021).
27. Van Hasselt, H., Guez, A., Silver, D.: Deep Reinforcement Learning with Double Q-Learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016).
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is All You Need. Advances in Neural Information Processing Systems **30** (2017).
29. Vezhnevets, A.S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., Kavukcuoglu, K.: FeUdal Networks for Hierarchical Reinforcement Learning. In: International Conference on Machine Learning, pp. 3540–3549. PMLR (2017).
30. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., De Freitas, N.: Dueling Network Architectures for Deep Reinforcement Learning. In: International Conference on Machine Learning, pp. 1995–2003. PMLR (2016).
31. Wang, Z., Liu, C., Gombolay, M.: Heterogeneous Graph Attention Networks for Scalable Multi-Robot Scheduling with Temporospatial Constraints. Autonomous Robots **46**(2), 249–268 (2022).
32. Williams, R.J.: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning **8**(3–4), 229–256 (1992).
33. Yan, M., Yuan, H., Xu, J., Yu, Y., Jin, L.: Task Allocation and Route Planning of Multiple UAVs in a Marine Environment Based on an Improved Particle Swarm Optimization Algorithm. EURASIP Journal on Advances in Signal Processing **2021**, 1–23 (2021).
34. Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, J., Xiong, R.: Multi-Robot Coordination with Spatial-Temporal Constraints in Dynamic Environments. IEEE Robotics and Automation Letters **4**(4), 3753–3760 (2019).