Robotics: Science and Systems 2026
Sydney, Australia, July 13–17, 2026

Event-Driven Sleep-Wake Scheduling for Heterogeneous Robots under LTL Constraints

Xuyang Li[1], Leilei Li[1], Jianwu Fang[1†], Boyuan Chen[1] and Jianru Xue[1†]
[1]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Institute of Artificial Intelligence and Robotics
Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
Email: {lixuyang,lileilei2024,2194111706}@stu.xjtu.edu.cn, {fangjianwu,jrxue}@mail.xjtu.edu.cn

***Abstract*—In large-scale heterogeneous robot systems (HRS), scheduling efficiency in terms of throughput and makespan relies on exploiting parallel execution, while human-issued safety and precedence instructions impose rigid temporal-logic constraints that create severe combinatorial complexity and challenge traditional optimization and metaheuristic methods. We propose Asynchronous Logic-Induced Sleep-Wake Coordination (ALIS-WC), an event-driven reinforcement learning (RL) framework that performs sleep-wake scheduling of heterogeneous robots under Linear Temporal Logic (LTL) constraints. At its core, ALIS-WC applies a two-stage filtering mechanism: it first filters out robots that are physically incapable of contributing to the current decision, and then uses temporal-logic dependencies to label the remaining robots as sleeping or eligible, so that the learned scheduling policy only coordinates among idle, constraint-compliant robots at each decision event. We first design a human-ALIS-WC interaction interface: a lightweight language front-end that maps natural-language (NL) mission descriptions into LTL safety and precedence clauses for the scheduler using a compact 8B translator that achieves high NL-to-LTL accuracy. We further design a potential-based reward-shaping term over normalized global mission progress that provides a difficulty-invariant learning signal and accelerates policy learning without changing the underlying objective. Experiments on large-scale graph-based benchmarks with dozens of heterogeneous robots, 100 tasks, and up to 10 LTL clauses show that ALIS-WC improves makespan and task completion over strong metaheuristic, search-based, and reinforcement-learning baselines while maintaining 100% satisfaction of all enforced temporal-logic constraints. Additional stress tests with up to 60 clauses indicate that ALIS-WC remains effective under much denser temporal-logic specifications. Code is publicly available at https://github.com/lixuyang-m/ALIS-WC.**

I. Introduction

With the deployment of large-scale, heterogeneous robot fleets in logistics, manufacturing, and service environments, multi-robot task allocation has become a central coordination challenge whose complexity scales sharply with fleet size and capability diversity [1]. Vehicle routing problems with multiple commodities represent a major logistics application of this paradigm, where heterogeneous fleets handle diverse cargo types across distributed depot networks [15]. Beyond optimizing efficiency, modern deployments frequently require externally imposed high-level instructions from human supervisors, such as priority adjustments for urgent shipments (e.g., medical supplies must be delivered before regular inventory) or temporary safety restrictions during maintenance (e.g., prohibiting vehicles from specific zones). These temporal relationships can be systematically formalized using Linear Temporal Logic (LTL) constraints, providing rigorous specification frameworks for complex operational requirements [27]. The combination of heterogeneous capabilities, multi-depot operations, and dynamic LTL specifications creates a complex scheduling scenario where fundamental task parallelizability conflicts with externally imposed sequential dependencies. This raises a critical question: **how can we achieve real-time scheduling for large-scale heterogeneous multi-robot systems while satisfying dynamic temporal logic constraints?**

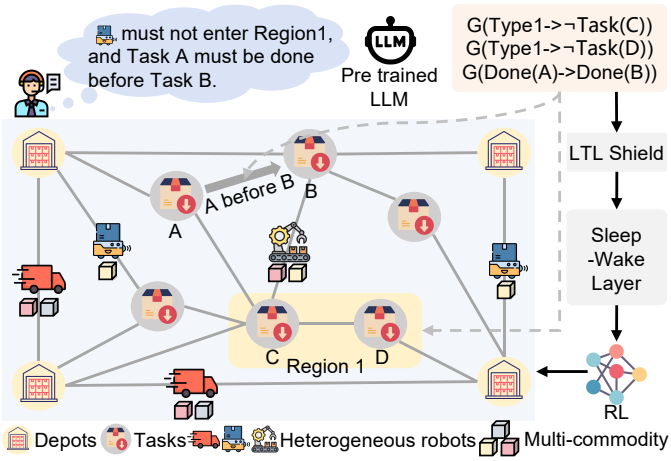

Fig. 1. Overview of the ALIS-WC framework in a heterogeneous multi-robot logistics scenario. Natural-language mission descriptions are optionally translated by an NL-to-LTL translator into LTL specifications, which feed an LTL-based shield and sleep–wake coordination layer that wrap a learned scheduling policy for heterogeneous robots in the logistics environment.

An illustrative heterogeneous logistics scenario with temporal-logic constraints is shown in Fig. 1. Addressing this scheduling problem requires balancing computational tractability with solution quality, leading to the following research directions. Traditional mathematical programming and metaheuristic approaches provide optimality guarantees and handle fine-grained capacity and inventory constraints

†Corresponding authors.
This work is supported by the NSFC (Grants No. W2411052 and 62273057), Outstanding Youth Foundation of Shaanxi Province (Grant No. 2025JC-JCQN-092), and Key Research and Development Program of Shaanxi (Grant No. 2025PT-ZCK-66).

effectively [19]. However, when LTL constraints are introduced, these methods suffer from exponential state-space explosion, making them unsuitable for real-time operations [33]. Reinforcement Learning (RL) frameworks have emerged as a promising alternative, offering generalizable policies and millisecond-level inference times. Existing RL approaches fall into two main streams: destination assignment for heterogeneous teams [10], and temporal-constraint-aware task scheduling [3]. These methods typically operate in simplified decision spaces or under static constraint structures, and do not explicitly address our dynamic LTL requirements. Additionally, existing LTL research in reinforcement learning predominantly focuses on single-robot scenarios [2] or small-scale multi-robot problems [11]. The fundamental challenge lies in the combinatorial explosion from coupling heterogeneous capabilities, multi-commodity loading decisions, and dynamic logical constraints, creating complexity that has not yet been addressed by existing RL architectures in large-scale heterogeneous settings with LTL constraints.

We address this challenge with an event-driven architecture that separates logical enforcement from large-scale heterogeneous scheduling through a sleep–wake coordination layer. At each decision event, an LTL-based shield masks out actions that would violate any active safety or precedence clause and temporarily puts logically blocked robots to sleep, so that only idle and constraint-compliant robots remain available for scheduling. The learned scheduling policy then focuses solely on coordinating among these currently available robots in the physical logistics environment, while the sleep–wake layer dynamically adapts to changing mission specifications. This plug-and-play design allows a single policy to operate under many different temporal-logic specifications without retraining, while maintaining strict logical compliance and high scheduling efficiency. Our main contributions are:

- We construct a unified event-driven platform that integrates heterogeneous robot capabilities, multi-commodity loading, multi-trip routing, and LTL temporal-logic constraints into a single discrete-event simulator and a single shielded action interface, enabling metaheuristic, search-based, and reinforcement-learning methods to be evaluated consistently under the same protocol.
- We introduce Asynchronous Logic-Induced Sleep-Wake Coordination (ALIS-WC), a modular event-driven architecture that decouples logical enforcement from physical coordination via a hard LTL-based shield and a two-stage sleep–wake layer, so that the learned scheduling policy only coordinates among idle, constraint-compliant robots.
- We implement ALIS-WC in a discrete-event, graph-based multi-robot logistics simulator and show that a single shielded policy consistently outperforms strong metaheuristic, search-based, and reinforcement-learning baselines in both unconstrained and LTL-constrained settings, while strictly satisfying all enforced temporal-logic constraints and supporting real-time deployment across a wide range of mission specifications without retraining.

II. Related Work

Heterogeneous multi-robot task allocation and scheduling. The heterogeneous MRTA problem has attracted significant research attention across optimization and learning paradigms. Traditional approaches leverage mixed-integer programming and metaheuristic methods [19], multi-objective formulations [7], branch-and-cut algorithms [33], and market-based mechanisms [20], but suffer from computational scalability limitations. Hybrid approaches bridge optimization with AI techniques through iterative clustering [25] and graph neural network enhanced ant colony optimization [24]. Learning-based approaches demonstrate strong generalization and real-time performance, including attention-based actor-critic frameworks [10], deep reinforcement learning for dynamic scheduling [35], heterogeneous graph transformers [3], attention mechanisms for vehicle routing [21], and specialized variants for long-endurance missions [8]. Our work builds on these foundations with an event-driven, attention-based scheduler that keeps logical enforcement external to the policy.

Temporal logic specifications in multi-robot systems. Linear Temporal Logic has been extensively studied for specifying high-level mission requirements in multi-robot coordination. Classical synthesis-based models generate feasible plans satisfying LTL specifications with hierarchical frameworks, enabling expressive temporal requirements [23] and providing real-time reactive capabilities [9]. Decomposition strategies manage computational complexity through parallel decomposition and concurrent satisfaction [37], while alternative approaches employ cross-entropy optimization [5]. Recent theoretical advances explore shield synthesis [28] and LTL synthesis under multi-robot assumptions [4]. Shield-RL [11] enforces LTL through action-level masks on grid-level movement, while TARGETNET [3] addresses one-shot robot-to-task direct-service allocation under temporal precedence constraints without multi-trip routing. These methods provide strong theoretical foundations but face computational challenges when scaling to large heterogeneous teams or handling dynamic constraints. On the natural-language side, general-purpose NL-to-LTL translators [12, 22] have been developed as standalone tools; we instead incorporate a domain-specialised compact translator as an optional front-end tightly coupled to our simulator's atomic propositions.

Graph neural networks and attention mechanisms for multi-robot coordination. Graph neural networks and attention mechanisms have emerged as powerful tools for handling complex relational structures in multi-robot coordination. They encode robot relationships, task dependencies, and environmental constraints into learnable structures, with examples including resilient distributed coordination under partial observability [13], multi-UAV tracking with dynamic communication graphs [36], and distributed RL with reward machines [18]. Our work also relies on attention-based coordination, but couples it with sleep–wake masking and an LTL shield to handle large-scale heterogeneous multi-robot scheduling problems.

III. METHODOLOGY

A. Problem Formulation

We consider a large-scale heterogeneous robot system operating in a planar workspace $\mathcal{W} \subset \mathbb{R}^2$ with a robot set $\mathcal{R} = \{r_1, \ldots, r_n\}$, a task set $\mathcal{T} = \{\tau_1, \ldots, \tau_m\}$, and a depot set $\mathcal{D} = \{d_1, \ldots, d_D\}$. Let Γ denote the set of cargo types. Each robot r_i has a capability vector $\mathbf{c}_i \in \mathbb{N}^{|\Gamma|}$ specifying its maximum capacity for each cargo type, so that different robots may specialize in different commodities or carry different loads. Each task τ_j is defined by $(p_j, \mathbf{r}_j)$, where p_j is its location and $\mathbf{r}_j \in \mathbb{N}^{|\Gamma|}$ is a multi-commodity demand vector.

Robots may execute *multi-trip* operation, *i.e.*, they can shuttle between depots and tasks to load and unload cargo, and a task is completed once its remaining demand becomes zero, potentially after multiple visits by multiple robots. This yields a combinatorial scheduling problem that generalizes multi-depot vehicle routing with multiple commodities and capacity constraints. An illustrative heterogeneous logistics scenario for this problem, together with the overall ALIS-WC scheduling architecture, is shown in Fig. 2. Therefore, we aim to obtain a feasible scheduling policy that coordinates all robots as they shuttle between depots and task locations to load and unload cargo, so that all task demands are met within the time limit.

Specifically, let $\mathbf{d}_j(t) \in \mathbb{R}_{\geq 0}^{|\Gamma|}$ denote the remaining demand vector of task τ_j at time t. We define the completion time of task τ_j under policy π as

$$C_j(\pi) := \inf\{t \geq 0 \mid \mathbf{d}_j(t) = \mathbf{0}\}. \tag{1}$$

The mission completion time (makespan) is

$$T(\pi) := \max_{\tau_j \in \mathcal{T}} C_j(\pi), \tag{2}$$

and our objective is to minimize the expected makespan:

$$\min_{\pi} \mathbb{E}_{\pi}[T(\pi)]. \tag{3}$$

B. LTL Specifications and Natural-Language Interface

In real-world deployments, minimizing makespan alone is insufficient, as missions must also satisfy high-level safety and precedence requirements. Human supervisors naturally state such rules in free-form language (*e.g.*, "*robots must never enter the maintenance zone*" or "*urgent orders must be completed before regular restocking*"), which we formalize as Linear Temporal Logic (LTL) clauses. We focus on two common families: *safety clauses* of the form "robot r must never visit node v" and *sequential clauses* of the form "task A must be completed before task B".

Inspired by recent work on translating natural language commands into LTL specifications [12, 22], we build a two-stage NL-to-LTL interface to avoid hand-writing LTL formulas for every mission. As illustrated in Fig. 3, we first automatically generate LTL clauses offline from a domain grammar and use a large general-purpose LLM (GPT-4.5[1]) to produce diverse natural-language descriptions, retaining human-verified NL–LTL pairs as a training corpus. We then fully fine-tune a compact 8B instruction-tuned model [34] on this corpus to obtain our deployed NL-to-LTL translator. At deployment time, operators enter mission rules in natural language, the translator proposes candidate LTL clauses for human confirmation, and the resulting set $\Phi = \{\varphi_1, \ldots, \varphi_C\}$ serves as hard temporal-logic constraints to be enforced on the action space at every decision epoch. This design keeps the human in the loop so that residual translation errors can be filtered before deployment.

[1]https://openai.com/index/introducing-gpt-4-5/

C. Event-Driven MDP Framework

We model the constrained scheduling problem, corresponding to the event-driven coordination block in Fig. 2, as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ evolving over decision epochs indexed by k, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space (MOVE to a node, LOAD cargo at a depot, or UNLOAD cargo at the current location), P the transition kernel (deterministic in our setting, induced by the discrete-event simulator), R the per-event reward function, and γ the discount factor; we work in the undiscounted setting $\gamma = 1$ throughout, justified by the finite-horizon episode structure. Let t_k denote the time of the k-th decision epoch, $x_k \in \mathcal{S}$ the Markov state, $a_k \in \mathcal{A}(x_k)$ the action of the scheduled robot, $x_{k+1} \sim P(\cdot \mid x_k, a_k)$ the next state, $r_k = R(x_k, a_k, x_{k+1})$ the per-epoch reward, and $\Delta_k = t_{k+1} - t_k$ the time interval until the next decision. The objective is to find a policy $\pi_\theta(a \mid x)$ with parameters θ that maximizes the expected return [30]

$$J(\pi) \;=\; \mathbb{E}_{\pi}\Big[\sum_{k=0}^{K-1} r_k\Big], \tag{4}$$

where K denotes the number of decision epochs in an episode (a random variable depending on π and the instance). We write $i_k \in \{1, \ldots, |\mathcal{R}|\}$ for the index of the robot dispatched at epoch k.

Event-driven coordination. As illustrated in Fig. 4, the system advances through discrete decision events rather than fixed time steps. A decision epoch t_k is triggered whenever any robot completes its current operation (e.g., robots ① and ② completing *Move* at t_1, or robot ③ completing *Load* at t_2). When multiple robots reach decision points simultaneously, such as at t_1, their decisions are processed in a random order (including the initial epoch $t_0 = 0$) to avoid systematic bias during training and prevent the policy from hard-coding a fixed processing order. Since heterogeneous operations (*Move*, *Load* and *Unload*) possess varying execution durations, the system clock advances asynchronously to the next earliest completion event, with each robot r carrying a timestamp $\mathrm{next_decision}(r)$ recording when it next becomes ready.

State space. The Markov state at decision epoch k is

$$x_k = (s_k^{\text{robot}}, s_k^{\text{task}}, s_k^{\text{depot}}), \tag{5}$$

where x_k collects the physical logistics variables for robots, tasks, and depots, all encoded ego-centrically with the currently scheduled robot r_{i_k} as the spatial origin (i.e., every

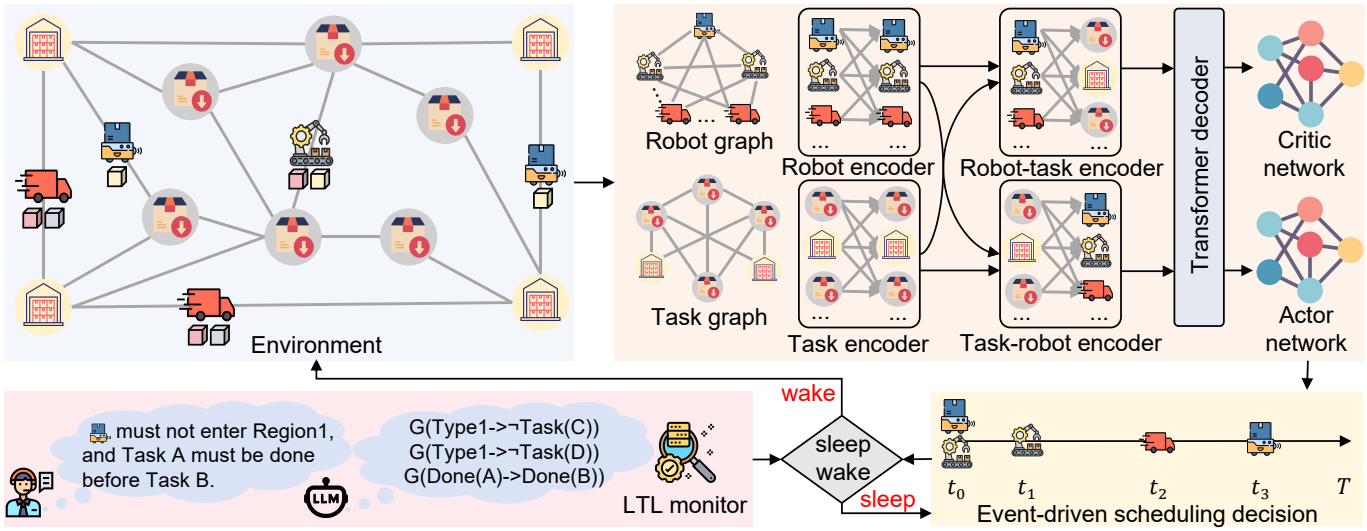

Fig. 2. Overall architecture of the ALIS-WC scheduler for LTL-constrained, event-driven multi-robot coordination. The top-left panel shows a heterogeneous logistics environment with robots, depots, and task locations. Natural-language mission descriptions are translated into formal LTL clauses and monitored by the LTL monitor, which enforces hard safety and sequencing constraints by masking out unsafe actions. Robot, task, and depot states are encoded as token sets processed by self- and cross-attention to produce structured features for a Transformer-based actor–critic policy. Along an asynchronous timeline of operation-completion events, this policy issues sleep–wake and dispatch decisions that coordinate robots while strictly satisfying the LTL specifications.

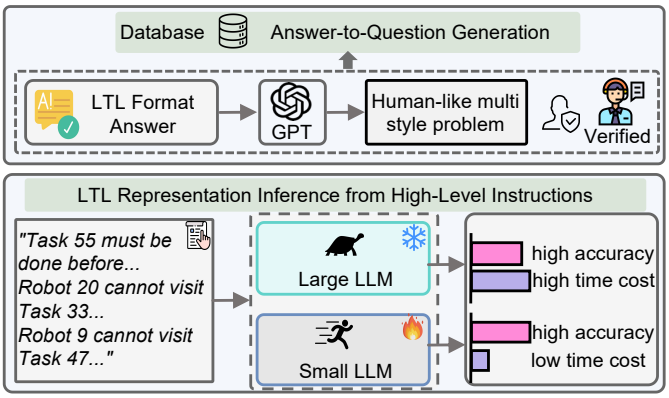

Fig. 3. Construction of the NL-to-LTL interface. A programmatic generator emits LTL clauses over the simulator's atomic propositions; a frozen general-purpose paraphraser (GPT-4.5) produces five paraphrase styles per clause (formal / casual / urgent / explanatory / brief), and human-verified pairs form the training corpus. A compact 8B instruction-tuned model (Qwen3-8B [34]) is fully fine-tuned on this corpus and deployed as the NL-to-LTL translator that converts operator commands into clauses for the ALIS-WC shield. Dataset, hyperparameters, and prompt templates are detailed in Suppl. Sec. I and IV.

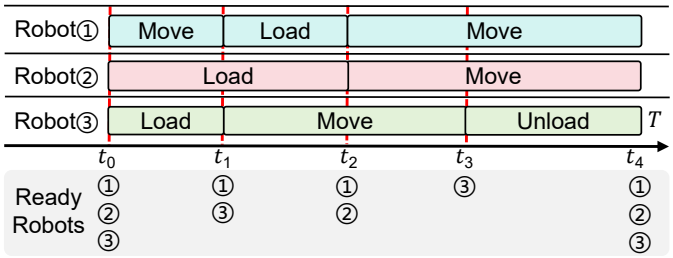

Fig. 4. Illustration of the asynchronous event-driven coordination. Decision epochs t_k are triggered by the completion of robot operations (e.g., *Move*, *Load*), resulting in variable temporal intervals between events.

offset $\Delta p_{i_k\to}$. below denotes a position relative to p_{i_k}).

$$s_k^{\text{robot}} = \left\{(\Delta p_{i_k\to m},\, \mathbf{inv}_m,\, \mathbf{cap}_m,\, \text{status}_m,\, \tau_m^{\text{rem}})\right\}_{m=1}^{|\mathcal{R}|}, \quad (6)$$

where $\Delta p_{i_k\to m} = p_m - p_{i_k} \in \mathbb{R}^2$ is the spatial offset from r_{i_k} to robot r_m, $\mathbf{inv}_m \in \mathbb{Z}_{\geq 0}^{|\Gamma|}$ is its current cargo inventory, $\mathbf{cap}_m \in \mathbb{Z}_{\geq 0}^{|\Gamma|}$ is its capacity vector (static within an episode but resampled across episodes from Table I, hence retained in the state to encode heterogeneity for cross-episode generalization), status_m is a discrete operational mode (idle, traveling, inactive, temporarily sleeping), and $\tau_m^{\text{rem}} \geq 0$ is the remaining travel time if r_m is en route (zero otherwise). In addition, we append two scalar time features to every robot feature vector: the *normalized current time* $t_k/T_{\max} \in [0,1]$ and the *remaining time* $(T_{\max}-t_k)/T_{\max} \in [0,1]$, where $T_{\max}$ is the per-episode time budget. The task-state component aggregates destination features for all tasks:

$$s_k^{\text{task}} = \left\{(\Delta p_{i_k\to j},\, \mathbf{r}_j^{\text{remain}},\, \mathbf{r}_j,\, \tau_{i_k\to j}^{\text{travel}},\, \tau_j^{\text{proc}},\, \text{status}_j)\right\}_{j=1}^{|\mathcal{T}|}, \quad (7)$$

where $\Delta p_{i_k\to j}$ is the offset from r_{i_k} to task τ_j, $\mathbf{r}_j^{\text{remain}} \in \mathbb{Z}_{\geq 0}^{|\Gamma|}$ is the remaining demand vector, $\mathbf{r}_j \in \mathbb{Z}_{\geq 0}^{|\Gamma|}$ is the original total demand, $\tau_{i_k\to j}^{\text{travel}}$ is the travel time from the current robot location, τ_j^{proc} is a per-task feature shown to the policy (the simulator applies a fixed LOAD/UNLOAD latency at every visit), and $\text{status}_j \in \{\text{active}, \text{completed}\}$ indicates completion.

The depot-state component uses analogous relative features:

$$s_k^{\text{depot}} = \left\{ (\Delta p_{i_k \to d},\, \mathbf{stock}_d,\, \boldsymbol{\rho}_d,\, \mathrm{id}_d^{\text{norm}}) \right\}_{d=1}^{|\mathcal{D}|}, \tag{8}$$

where $\Delta p_{i_k \to d}$ is the offset from r_{i_k} to depot d, $\mathbf{stock}_d$ denotes current stock levels normalized by the capacities of the scheduled robot r_{i_k} ("how many full loads" of r_{i_k} remain per commodity), $\boldsymbol{\rho}_d$ is the element-wise stock ratio between current and initial stock, and $\mathrm{id}_d^{\text{norm}} \in [0, 1]$ is a normalized depot identifier.

Action space. The action space $\mathcal{A}$ is composite and discrete: at each decision epoch, the policy first samples an operation type from $\{\text{MOVE}, \text{LOAD}, \text{UNLOAD}\}$, then, depending on this choice, selects additional discrete arguments. If MOVE is selected, the policy further samples a destination node (task or depot) from a masked categorical distribution. If LOAD or UNLOAD is selected, the policy instead samples a cargo type.

Policy optimization. We train the shared-parameter actor–critic with Proximal Policy Optimization (PPO) [29] on the factorized policy over operation type, destination, and cargo type for the scheduled robot r_{i_k}.

Sleep–wake coordination for zero-shot LTL generalization. Let $\mathcal{E}$ denote the discrete-event simulator, $\mathcal{M}$ the LTL monitor (a finite-state automaton, FSA, over Φ) that tracks the satisfaction state of each clause, and $r \in \mathcal{R}$ a generic robot; an FSA transition refers to a state change of the automaton of some clause in Φ, typically triggered by the completion of a predecessor task. The masking pipeline of Algorithm 1 (lines 8–11) is built around two concerns. First, ALLOCATIONFILTER deducts the in-flight cargo reservations of simultaneously-ready peers from the residual demand visible to robot r, so that no two robots commit to the same cargo unit at one epoch. Second, FEASIBILITYMASK enforces all *permanent* action restrictions: physical feasibility (capacity, inventory, reachability) together with the state-independent LTL safety clauses in Φ. After this stage, a robot's surviving action set is the set of actions through which it could in principle contribute to the global residual demand. The final stage, SEQUENTIALSHIELD, enforces the *temporary* restrictions imposed by sequential LTL clauses against the current FSA state of $\mathcal{M}$. The robot's status at this epoch (lines 12–18) follows directly from where in the pipeline its action set was emptied: when the final set is non-empty, π_θ samples its action (and clears any sleep status); when the post-feasibility action set is already empty, the robot has no contribution under the permanent restrictions and is marked *inactive* for the rest of the episode; when only SEQUENTIALSHIELD was responsible for the emptying, the robot is set to *temporarily sleeping* and re-admitted to the ready queue (lines 5–6) the next time an FSA transition might unblock it. Since π_θ only ever conditions on the post-mask action set, training (where $\Phi = \emptyset$ makes the safety part of FEASIBILITYMASK trivial and SEQUENTIALSHIELD a no-op) and evaluation under arbitrary Φ present identical interfaces to the policy, and the same θ generalizes zero-shot to any clause set.

Sparse base reward. Each episode terminates either successfully when all tasks are completed within the time limit

Algorithm 1 Event-driven scheduling with sleep–wake under LTL shield

Require: environment $\mathcal{E}$, policy π_θ, LTL monitor $\mathcal{M}$ over clauses Φ

1: **while** $\mathcal{E}$ is not terminated **do**
2: $t \leftarrow$ next decision time over all robots; advance $\mathcal{E}$
3: $F \leftarrow$ tasks completed at t; update $\mathcal{M}$ on each $\tau_j \in F$
4: $Q \leftarrow \{r \in \mathcal{R} : r \text{ ready at } t\}$
5: **if** any $\tau_j \in F$ triggers an FSA transition in $\mathcal{M}$ **then**
6: $Q \leftarrow Q \cup \{r \in \mathcal{R} : r \text{ is temporarily sleeping}\}$
7: **end if**
8: **for** $r \in \text{shuffle}(Q)$ **do** ▷ r is the scheduled robot
9: $A \leftarrow \text{ALLOCATIONFILTER}(r, \mathcal{E})$
10: $A \leftarrow \text{FEASIBILITYMASK}(A, r, \mathcal{E}, \mathcal{M})$ ▷ physical + LTL safety (permanent)
11: $A_{\text{cap}} \leftarrow A$ ▷ post-feasibility action set
12: $A \leftarrow \text{SEQUENTIALSHIELD}(A, \mathcal{M})$ ▷ LTL sequential (temporary)
13: **if** $A \neq \emptyset$ **then**
14: **if** r was sleeping **then**
15: $\text{next_decision}(r) \leftarrow t$; clear r's sleep status
16: **end if**
17: sample $a \sim \pi_\theta(\cdot \mid x_k, r)$ under mask A; $\mathcal{E}$.execute(r, a)
18: **else if** $A_{\text{cap}} = \emptyset$ **then**
19: mark r as *inactive* (permanent)
20: **else**
21: mark r as *temporarily sleeping*
22: **end if**
23: **end for**
24: **end while**

$T_{\max}$, or with failure if the time limit is exceeded while some tasks remain unfinished. We use a sparse base reward

$$r_{\text{episode}} = \begin{cases} -T_{\text{current}} & \text{if all tasks completed before time limit} \\ -T_{\max} & \text{if time limit exceeded (failure),} \end{cases} \tag{9}$$

where T_{current} is the mission completion time of the episode. We assign $r_k = 0$ for all $k < K-1$ and $r_{K-1} = r_{\text{episode}}$ at the terminal step, so that $J(\pi) = \mathbb{E}_\pi[r_{\text{episode}}]$ directly minimises the makespan on successful episodes.

Potential-based shaping. The base reward is event-sparse—non-zero only at termination—so we add a potential-based shaping term [26] over normalized global mission progress. Let $\psi(x_k) \in [0, 1]$ denote the fraction of tasks completed by state x_k relative to the total task count, with $\psi(x_0) = 0$ at the beginning of each episode and $\psi(x_K) = 1$ whenever all tasks are completed. The shaped reward at each step is $\tilde{r}_k = r_k + \lambda(\psi(x_{k+1}) - \psi(x_k))$ with $\lambda = 10$ in all experiments, and the per-episode shaped return telescopes to

$$\begin{aligned} \sum_{k=0}^{K-1} \tilde{r}_k &= \sum_{k=0}^{K-1} \big(r_k + \lambda(\psi(x_{k+1}) - \psi(x_k))\big) \\ &= r_{\text{episode}} + \lambda\psi(x_K). \end{aligned} \tag{10}$$

For successful episodes, where $\psi(x_K) = 1$, this evaluates to $-T_{\text{current}} + \lambda$, which is strictly monotone in the makespan up to an additive constant; the shaped objective therefore agrees with the base objective on the set of successful policies. For failed episodes, where $\psi(x_K) \in [0, 1)$, the residual term $\lambda\psi(x_K)$ provides a dense gradient that rewards trajectories completing more of the global demand before failure, accelerating policy learning in the early phase of training without altering the optimal policy [26].

D. Neural Network Architecture

We adopt a shared-parameter attention-based actor–critic network corresponding to the central policy module in Fig. 2. At each decision epoch k, the network receives the global state defined in Sec. III-C and outputs a robot-conditioned policy $\pi_\theta(\cdot \mid x_k, i_k)$ for the scheduled robot r_{i_k} together with a value estimate $V_\theta(x_k, i_k)$. The policy is trained end-to-end with PPO on a diverse set of unconstrained training instances without LTL clauses or sleep–wake logic. At deployment time, we activate the LTL shield and the masking-and-sleep–wake mechanism described in Sec. III-C while keeping the policy parameters fixed, enabling zero-shot transfer from unconstrained training to LTL-constrained operation.

Task–robot attention backbone. The backbone encodes the physical state with set-based attention: we embed robot and destination features, apply self-attention within each set, and use cross-attention blocks to exchange information between destinations and robots. Concretely, the robot, task, and depot feature sets $s_k^{\text{robot}}, s_k^{\text{task}}, s_k^{\text{depot}}$ are first mapped to embedding matrices $H_k^{\text{robot}}, H_k^{\text{task}}, H_k^{\text{depot}}$ and passed through several layers of multi-head self-attention [31] within each set to produce $\tilde{H}_k^{\text{robot}}, \tilde{H}_k^{\text{task}}, \tilde{H}_k^{\text{depot}}$. We then apply cross-attention from robots to tasks,

$$\hat{H}_k^{\text{robot}} = \text{Attn}\big(Q = \tilde{H}_k^{\text{robot}},\, K = \tilde{H}_k^{\text{task}},\, V = \tilde{H}_k^{\text{task}}\big), \quad (11)$$

so that each robot representation aggregates information from relevant tasks and depots. A pointer-style conditioning mechanism then uses the current robot index i_k to query the robot sequence and obtain an embedding h_{i_k} summarizing what the scheduled robot "sees" in the global state.

Policy and value heads. The actor head factorizes the action into (i) an operation-type distribution over {MOVE,LOAD,UNLOAD}, (ii) a destination pointer for MOVE, and (iii) cargo-type choices for LOAD/UNLOAD, with hard masking applied to invalidate infeasible options. Formally, letting o_k, j_k, ℓ_k denote the operation type, destination, and cargo type at epoch k, the policy factorizes as $\pi_\theta(a_k \mid x_k, i_k) = \pi_\theta^{\text{op}}(o_k \mid h_{i_k})\, \pi_\theta^{\text{dest}}(j_k \mid o_k, h_{i_k})\, \pi_\theta^{\text{cargo}}(\ell_k \mid o_k, j_k, h_{i_k})$, where each factor is implemented as a masked categorical distribution. For LOAD/UNLOAD operations, the environment then deterministically applies the maximum admissible quantity given the remaining demand, inventory and capacity constraints. The critic head outputs a scalar value estimate $V_\theta(x_k, i_k)$ for the scheduled robot.

IV. EXPERIMENTS

In this section we empirically evaluate ALIS-WC and its components. Our experiments are designed to answer the following questions:

- **(Q1)** How well does ALIS-WC perform compared to strong optimization and heuristic baselines in the unconstrained (no-LTL) setting?
- **(Q2)** Can a single policy trained without LTL constraints be deployed zero-shot under hard LTL specifications drawn from the safety and sequential families while maintaining performance and LTL satisfaction?
- **(Q3)** How much does progress-based potential shaping improve sample efficiency and final performance over a purely sparse reward?
- **(Q4)** Is the natural-language (NL)-to-LTL interface accurate and fast enough for interactive, real-time use?

We first describe the common experimental setup, then evaluate the NL-to-LTL front-end, and report the main scheduling results with and without LTL constraints. The effect study of reward shaping is finally presented.

A. Experimental Setup

We implement all methods in a custom discrete-event simulator for the heterogeneous multi-depot, multi-commodity setting. Each training scenario is generated by randomly sampling fleet composition, depot layout, task set, and cargo demands within the ranges in Table I, so that the policy is exposed to a broad spectrum of problem complexities. The simulator advances time according to the asynchronous event-driven dynamics and executes primitive MOVE, LOAD and UNLOAD operations with deterministic effects on locations, inventories and task demands, and we always measure makespan as the time when all task demands become zero. No LTL clauses are attached during training; LTL constraints are introduced only at evaluation time when constructing the LTL-constrained benchmarks in Tables III and IV.

Figure 5 illustrates a representative benchmark instance in our normalized 2D environment. Colored circles (e.g., R0–R8) denote heterogeneous robots with different load capacities, red triangles are depots, and squares are tasks. Dashed arrows show current robot routes, while solid arrows and "blocked" overlays indicate LTL precedence and safety constraints that temporarily disable some tasks.

Baselines. We compare ALIS-WC against four baselines, grouping them by their planning regime: *Offline* methods (GA, AVNR) solve a batch optimization over the entire problem instance before execution begins; *Online* methods (Greedy, ALIS-WC) instead receive the current environment state at every decision epoch and issue one decision per event.

AVNR. An adaptive variable-neighborhood search (AVNR) inspired by adaptive large neighborhood search for dynamic VRPs [32]. It repeatedly applies destroy-and-repair operators with adaptive weight updates to improve a completion-time objective on multi-depot routing instances. We adapt its cost function and feasibility checks to match our makespan criterion and basic capacity constraints.

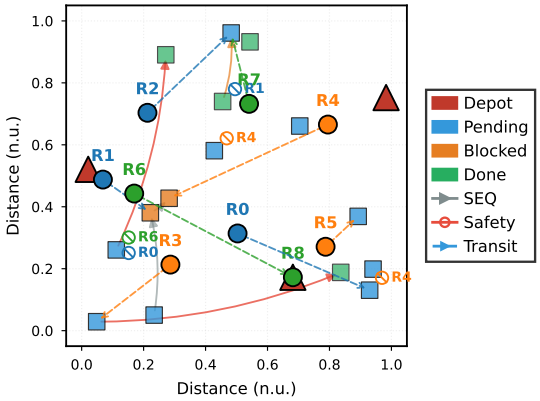

Fig. 5. Example benchmark instance in the 2D simulator. Circles (R0–R8) denote robots, triangles are depots, and squares are tasks; dashed paths show current robot routes, while solid arrows with "blocked" markers indicate tasks temporarily disabled by LTL safety and precedence constraints.

TABLE I
TRAINING SCENARIO PARAMETER RANGES. ALL TASKS SHARE A COMMON SERVICE TIME $\tau_j^{\text{PROC}} = \tau_{\text{LOAD}}$.

Parameter	Symbol	Range / value
robot species	N_{sp}	3–6
robots per species	N_{veh}	3–6
Total robots	$\lvert\mathcal{R}\rvert$	9–36
Depots	$\lvert\mathcal{D}\rvert$	3–6
Tasks	$\lvert\mathcal{T}\rvert$	15–100
Cargo types	$\lvert\Gamma\rvert$	5
Max task demand	$d_{\max}$	2
Max robot capacity	$c_{\max}$	5
Load/unload duration	τ_{load}	0.2
Episode time limit	$T_{\max}$	500

GA. A genetic algorithm (GA) [19]. GA encodes multi-trip routes for all robots in a single chromosome and improves them through selection, crossover, and mutation. We adapt its objective to our makespan criterion and enforce the same basic capacity and fleet-size limits as in our simulator.

Greedy. (ours) Built on top of our discrete-event simulator, LTL shield, and sleep–wake masking interface. At each decision epoch, Greedy observes the masked feasible actions produced by the environment and shield and selects actions according to a hand-crafted scoring rule that prioritizes the shortest travel time among feasible actions.

ALIS-WC (ours). Our ALIS-WC scheduler instantiates the event-driven PPO policy described in Sec. III on the full heterogeneous multi-depot, multi-commodity problem. The policy is trained only in the unconstrained environment and evaluated without retraining in both no-LTL and LTL-constrained settings using the shield and sleep–wake mechanism.

RL baseline (sparse). [10] As a strong RL baseline, we use the same event-driven PPO algorithm and network architecture as ALIS-WC but remove the progress-based shaping term, training this variant under a purely sparse task-completion reward. This mirrors recent attention-based RL schedulers for simplified routing and scheduling problems and allows us to isolate the effect of reward shaping in our more challenging heterogeneous setting.

Evaluation metrics. We assess system performance using five key metrics that capture both efficiency and constraint satisfaction. (1) *Success rate (SR)* is the fraction of episodes that finish all tasks before the time limit while satisfying all LTL clauses (if any); in the no-LTL setting this reduces to requiring only timely task completion. (2) *Task completion rate (Task Fini. Rate)* is the average fraction of tasks completed across all evaluation episodes. (3) *Makespan* is the average mission completion time in successful episodes, directly matching our primary objective. (4) *Travel distance* is the total Euclidean distance traveled by all robots, reflecting operational cost. (5) *Solving time* is the average decision-making time per episode in seconds, characterizing real-time applicability. (6) *overall LTL satisfaction rate*, defined as the fraction of temporal-logic clauses satisfied across all evaluated episodes.

Training configuration. All RL training runs are conducted on a workstation equipped with an NVIDIA RTX 4090 GPU, we train the PPO for 10M environment steps with mini-batch Adam and standard PPO hyperparameters. The NL-to-LTL translator is trained on 8 A100 GPUs, using Qwen3-8B with either full-parameter or LoRA adaptation and a batch size of 1 with gradient-accumulation steps of 4 on the corpus.

Adaptation and evaluation protocol of the baselines. We adapt each baseline to our setting and report all metrics in Table IV from a single discrete-event simulator with the same LTL shield. *GA* encodes the multi-trip route sequences of all robots as a chromosome; at every generation, each individual's plan is rolled out in our discrete-event simulator, tasks blocked by the LTL shield are skipped during the rollout, and the resulting drop in completion rate is added to the fitness as a hard penalty. *AVNR* scores thousands of candidate neighbours per iteration, which is prohibitive if the simulator is invoked on every candidate; AVNR therefore uses an analytical multi-trip completion-time formula inside its search (accumulating depot-to-depot travel times and fixed load/unload times along each robot's trip sequence and taking the maximum across robots as the makespan), and the final returned solution is then passed through the *same* shielded simulator used by the other methods to produce the metrics reported in Table IV. *Greedy* and *ALIS-WC* both run inside the simulator, querying the three-mask pipeline (ALLOCATIONFILTER, FEASIBILITYMASK, SEQUENTIALSHIELD) at every decision epoch and sharing the same sleep–wake mechanism; the values reported in the tables are direct observations of this rollout.

B. Natural-Language Specification Evaluation

Task and dataset. We first construct an NL–LTL corpus and train the translator as a human-in-the-loop NL–LTL interface for mission specification. The NL–LTL corpus and data-generation pipeline follow Sec. III-B and Fig. 3: programmatically generated LTL clauses over the simulator's atomic propositions are paired with diverse paraphrases from large language models, yielding about 40k candidate NL–LTL pairs. From this pool we sample 6k pairs for training and hold out 4k test pairs (2k simple, 2k hard). The *simple* subset contains short, unambiguous descriptions, whereas the *hard*

TABLE II
ACCURACY AND DECODING TIME FOR NL-TO-LTL TRANSLATION ON THE SIMPLE AND HARD TEST SUBSETS. EACH REPORTS ACCURACY / TIME.

Method	Simple	Hard
Qwen3-8B (full FT) [34]	**99.5%** / **0.87s**	**92.4%** / 7.41s
Qwen3-8B (LoRA) [34, 17]	59.2% / 1.94s	16.0% / 7.39s
Qwen3-4B-Instruct [34]	39.1% / 0.96s	0.75% / 9.16s
Qwen3-30B-A3B-Instruct [34]	53.8% / 5.26s	4.85% / 36.9s
Llama3-8B [14]	9.75% / 6.40s	0.55% / 6.17s
Internlm3-8B-instruct [6]	29.9% / 9.66s	0.75% / 9.51s
Deepseek-R1-Dis8B [16]	40.3% / 0.92s	0.15% / **5.41s**
Deepseek-R1-Dis14B [16]	41.3% / 27.9s	0.75% / 10.2s
Mistral-Small-3.1-24B-Instruct[2]	46.2% / 1.08s	17.5% / 9.10s

subset consists of longer multi-clause commands with task dependencies and human-like distractor phrases.

Models. We compare a domain-adapted Qwen3-8B translator [34] against several open-weight instruction-tuned models. For Qwen3-8B we use both full-parameter and lightweight LoRA adaptation [17]. Additional off-the-shelf models include Qwen3-4B/30B [34], Mistral-Small-3.1-24B-Instruct, DeepSeek-R1-8B/14B [16], Llama3-8B [14], and InternLM3-8B [6], providing a representative sample of general-purpose instruction-tuned LLMs.

Metrics. We report exact-match accuracy between predicted and ground-truth LTL formulas after normalizing for syntactic variations, separately on the simple and hard subsets. To characterize the computational footprint of the interface, we also log decoding time (seconds per command) as a proxy for latency and resource usage.

Results. Table II summarizes the performance of these models. The fully fine-tuned Qwen3-8B translator, which we use in our deployed interface, attains 99.5% accuracy on the simple subset and remains above 90% on the hard subset, whereas all off-the-shelf models perform dramatically worse, often below 20% on hard commands. The LoRA-adapted Qwen3-8B improves over generic baselines but still trails full fine-tuning by a large margin. From a systems perspective, these results justify our choice of a fully fine-tuned 8B translator: it provides sufficiently high accuracy to avoid NL-to-LTL becoming the dominant error source, while maintaining acceptable decoding latency for interactive use.

C. Benchmark Results

We organise the evaluation into four difficulty *tiers*, each corresponding to one row block of Tables III and IV and to one fleet-size / depot-count / task-count configuration from Table I; tier 1 is the smallest instance and tier 4 the largest, with the number of safety and sequential clauses (C_s, C_{seq}) (where $C = C_s + C_{\text{seq}}$) scaling accordingly under LTL. To answer Q1, Table III compares ALIS-WC with GA, AVNR and a Greedy heuristic on the unconstrained benchmarks. Greedy and ALIS-WC achieve 100% success and task completion across all tiers, whereas GA and AVNR already struggle on the larger instances. In terms of efficiency, ALIS-WC consistently attains the shortest makespan and travel distance with modest online decision time, with Greedy ranking second and the offline methods trailing behind despite long per-instance budgets. As the problem scale grows from tier1 to tier4, the gaps in makespan and travel distance between ALIS-WC and the baselines widen, suggesting that ALIS-WC exploits parallelism and resource sharing opportunities that are difficult to capture with handcrafted heuristics or offline search.

To answer Q2, Table IV evaluates the same ALIS-WC policy when hard LTL constraints and the sleep–wake mechanism are activated. ALIS-WC maintains 100% success and LTL satisfaction across all tiers, with Greedy also satisfying all clauses but incurring systematically larger makespans and travel distances. In contrast, GA and AVNR experience severe drops in both success rate and LTL satisfaction, highlighting the advantage of learning-based scheduling under strict temporal-logic constraints. Furthermore, GA and AVNR require much longer per-instance solving times, whereas ALIS-WC (and Greedy) maintain millisecond-level decision latency, making them far more suitable for real-time deployment.

At the hardest tier we also run a stress test with 30 safety clauses and 30 sequential clauses to each instance from the families in Sec. III-B and reevaluating Greedy and ALIS-WC over three random seeds. As summarized in Table V, both methods degrade relative to Table IV, but Greedy's makespan worsen much more than those of ALIS-WC, indicating that the shielded ALIS-WC policy is more robust to dense temporal-logic constraints than a hand-crafted heuristic.

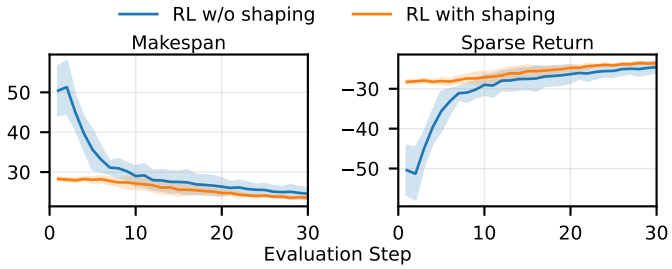

Fig. 6. Learning curves for RL with and without progress-based potential shaping on a fixed evaluation set. Left: average makespan; right: average episode return under the sparse task-completion reward. Curves show mean ± standard deviation over three random seeds.

D. Ablation: Reward Shaping

We ablate the progress-based potential shaping term by comparing our full variant to a sparse-reward baseline under identical architecture and training settings at the hardest benchmark tier, in both no-LTL and LTL-constrained benchmarks. Both policies are periodically evaluated on a fixed hold-out set. Figure 6 shows that shaping accelerates convergence and stabilizes training, and Table VI confirms that, at convergence, it yields shorter makespans and travel distances in both settings without reducing success rate. This behavior matches potential-based shaping theory: the optimal policy under the original return is unchanged, while the shaped reward provides denser learning signals that improve sample efficiency in our sparse-reward setting.

[2]https://mistral.ai/news/mistral-small-3-1

TABLE III

RESULTS IN THE NO-LTL SETTING ACROSS FOUR DIFFICULTY TIERS. EACH ENTRY REPORTS MEAN ± STANDARD DEVIATION OVER MULTIPLE RUNS.

Difficulty		Method	Mode	No LTL constraints				
				Succ. Rate.	Task Fini. Rate	Makespan	Trav. Dist.	Solv. Time
$\lvert\mathcal{D}\rvert = 3$	$N_{\text{sp}} = 3$	GA [19]	Offline	95.56%±1.92%	99.48%±0.34%	33.67±1.19	37.26±0.43	30.08±0.01
$N_{\text{veh}} = 3$	$\lvert\mathcal{T}\rvert = 15$	AVNR [32]		**100.00%±0.00%**	**100.00%±0.00%**	25.67±0.19	31.67±0.23	30.11±0.02
		Greedy	Online	**100.00%**	**100.00%**	22.75	32.63	**0.17**
		ALIS-WC		**100.00%±0.00%**	**100.00%±0.00%**	**19.97±0.40**	**29.67±0.13**	4.17±0.26
$\lvert\mathcal{D}\rvert = 4$	$N_{\text{sp}} = 4$	GA [19]	Offline	85.56%±5.09%	99.36%±0.47%	56.83±2.10	95.64±0.54	60.31±0.02
$N_{\text{veh}} = 4$	$\lvert\mathcal{T}\rvert = 40$	AVNR [32]		98.89%±1.92%	99.97%±0.05%	42.71±0.81	89.48±0.63	62.06±0.11
		Greedy	Online	**100.00%**	**100.00%**	29.64	74.42	**0.77**
		ALIS-WC		**100.00%±0.00%**	**100.00%±0.00%**	**25.57±0.22**	**63.97±0.82**	10.96±0.10
$\lvert\mathcal{D}\rvert = 5$	$N_{\text{sp}} = 5$	GA [19]	Offline	84.44%±8.39%	99.68%±0.11%	55.31±1.28	137.63±0.43	120.67±0.09
$N_{\text{veh}} = 5$	$\lvert\mathcal{T}\rvert = 65$	AVNR [32]		96.67%±3.33%	99.93%±0.08%	41.88±0.49	135.38±0.55	131.67±0.56
		Greedy	Online	**100.00%**	**100.00%**	25.58	98.53	**1.83**
		ALIS-WC		**100.00%±0.00%**	**100.00%±0.00%**	**21.45±0.19**	**83.08±1.03**	18.26±0.32
$\lvert\mathcal{D}\rvert = 6$	$N_{\text{sp}} = 6$	GA [19]	Offline	78.89%±7.70%	99.63%±0.26%	57.11±0.55	197.67±3.94	241.39±0.09
$N_{\text{veh}} = 6$	$\lvert\mathcal{T}\rvert = 100$	AVNR [32]		93.33%±3.33%	99.90%±0.07%	44.16±1.16	200.25±2.08	265.11±0.39
		Greedy	Online	**100.00%**	**100.00%**	24.78	156.57	**3.78**
		ALIS-WC		**100.00%±0.00%**	**100.00%±0.00%**	**20.03±0.26**	**114.01±1.39**	26.06±0.13

TABLE IV

RESULTS WITH LTL CONSTRAINTS ACROSS FOUR DIFFICULTY TIERS. EACH ENTRY REPORTS MEAN ± STANDARD DEVIATION OVER MULTIPLE RUNS.

Difficulty		Method	With LTL constraints					
			Succ. Rate.	Task Fini. Rate	Makespan	Trav. Dist.	Solv. Time	LTL Sat. Rate
$\lvert\mathcal{D}\rvert = 3$	$N_{\text{sp}} = 3$	GA [19]	44.44%±3.85%	93.41%±0.84%	38.17±0.63	34.48±1.15	30.07±0.01	78.06%±1.73%
$N_{\text{veh}} = 3$	$\lvert\mathcal{T}\rvert = 15$	AVNR [32]	0.00%±0.00%	84.74%±1.12%	–	–	30.12±0.04	52.78%±1.27%
$C_s = 2$	$C_{\text{seq}} = 2$	Greedy	**100.00%**	**100.00%**	23.15	36.74	**0.18**	**100.00%**
		ALIS-WC	**100.00%±0.00%**	**100.00%±0.00%**	**21.18±0.31**	**33.37±0.84**	5.45±0.42	**100.00%±0.00%**
$\lvert\mathcal{D}\rvert = 4$	$N_{\text{sp}} = 4$	GA [19]	7.78%±1.92%	94.92%±0.29%	60.06±12.06	102.10±7.20	60.28±0.02	69.44%±1.11%
$N_{\text{veh}} = 4$	$\lvert\mathcal{T}\rvert = 40$	AVNR [32]	0.00%±0.00%	92.14%±0.32%	–	–	62.08±0.46	52.78%±0.56%
$C_s = 3$	$C_{\text{seq}} = 3$	Greedy	**100.00%**	**100.00%**	31.54	86.51	**0.85**	**100.00%**
		ALIS-WC	**100.00%±0.00%**	**100.00%±0.00%**	**27.04±0.41**	**74.51±1.64**	12.14±0.24	**100.00%±0.00%**
$\lvert\mathcal{D}\rvert = 5$	$N_{\text{sp}} = 5$	GA [19]	1.11%±1.92%	95.79%±0.64%	52.45±0.00	132.65±0.00	120.67±0.05	68.89%±4.46%
$N_{\text{veh}} = 5$	$\lvert\mathcal{T}\rvert = 65$	AVNR [32]	0.00%±0.00%	93.81%±0.41%	–	–	133.58±0.85	53.61%±1.20%
$C_s = 4$	$C_{\text{seq}} = 4$	Greedy	**100.00%**	**100.00%**	25.99	109.32	**1.95**	**100.00%**
		ALIS-WC	**100.00%±0.00%**	**100.00%±0.00%**	**22.27±0.32**	**92.72±0.76**	19.52±0.47	**100.00%±0.00%**
$\lvert\mathcal{D}\rvert = 6$	$N_{\text{sp}} = 6$	GA [19]	2.22%±1.92%	96.61%±0.32%	55.85±10.95	189.50±41.32	241.61±0.12	71.33%±1.76%
$N_{\text{veh}} = 6$	$\lvert\mathcal{T}\rvert = 100$	AVNR [32]	0.00%±0.00%	95.07%±0.15%	–	–	265.14±1.00	54.00%±0.88%
$C_s = 5$	$C_{\text{seq}} = 5$	Greedy	**100.00%**	**100.00%**	25.84	156.57	**4.08**	**100.00%**
		ALIS-WC	**100.00%±0.00%**	**100.00%±0.00%**	**21.19±0.19**	**124.95±0.77**	29.88±0.69	**100.00%±0.00%**

TABLE V

STRESS-TEST RESULTS AT TIER 4 WITH 60 LTL CONSTRAINTS (C_s, C_{seq}). ALIS-WC RESULTS ARE AVERAGED OVER THREE SEEDS; GREEDY IS DETERMINISTIC UNDER THE FIXED INSTANCE DISTRIBUTION.

Method	Succ. Rate	Makespan	Trav. Dist.	LTL Sat. Rate
Greedy	**100%**	29.96	185.51	**100%**
ALIS-WC	**100%**	**26.28**±0.36	**160.33**±1.02	**100%**

TABLE VI

EFFECT OF POTENTIAL SHAPING ON ALIS-WC (DENOTED "OURS") AT THE HARDEST BENCHMARK TIER.

Setting	Variant	SR	Makespan	Trav. Dist.
No LTL	Ours w/o shap. [10]	**100%**	20.82 ±1.57	122.85 ±9.99
	Ours + shap.	**100%**	**20.03±0.26**	**114.01±1.39**
With LTL	Ours w/o shap. [10]	**100%**	22.04 ±1.19	133.08 ±9.72
	Ours + shap.	**100%**	**21.19±0.19**	**124.95±0.77**

V. CONCLUSION

We presented ALIS-WC, an event-driven RL scheduler for heterogeneous multi-robot logistics missions with high-level temporal specifications. Trained only in unconstrained simulations, a single policy combined at deployment with the sleep–wake coordination mechanism generalizes zero-shot to LTL-constrained deployments, strictly respects all specified temporal-logic clauses, and outperforms a range of strong metaheuristic, search-based, and RL baselines. **Across all evaluated tiers and clause densities, ALIS-WC scales to all instances without retraining.** Our experiments evaluate ALIS-WC in a graph-based multi-robot logistics simulator with two representative families of LTL constraints, covering safety and sequencing clauses. Extending the framework to richer classes of temporal constraints (such as liveness, periodic, or reactive specifications, which would require the policy to condition on the full automaton state rather than only on clause-completion events) and to additional operational objectives is an important direction for future work.

REFERENCES

[1] Athira A, Divya J, and Umashankar Subramaniam. A systematic literature review on multi-robot task allocation. *ACM Computing Surveys*, 57, 10 2024.

[2] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI*, pages 2669–2678, 2018.

[3] Batuhan Altundas, Shengkang Chen, Shivika Singh, Shivangi Deo, Minwoo Cho, and Matthew Craig Gombolay. Heteroge-

neous graph transformers for simultaneous mobile multi-robot task allocation and scheduling under temporal constraints. In *NeurIPS*, 2025.

[4] Benjamin Aminof, Giuseppe De Giacomo, Giuseppe Perelli, and Sasha Rubin. LTL Synthesis Under Multi-Agent Environment Assumptions. In *KR*, pages 719–728, 10 2025.

[5] Christopher Banks, Sean Wilson, Samuel Coogan, and Magnus Egerstedt. Multi-agent task allocation using cross-entropy temporal logic optimization. In *ICRA*, pages 7712–7718, 2020.

[6] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

[7] Tiziana Calamoneri, Federico Corò, and Simona Mancini. A matheuristic for multi-depot multi-trip vehicle routing problems. In *MIC*, volume 13838 of *Lecture Notes in Computer Science*, pages 464–469, 2022.

[8] Alvaro Calvo and Jesús Capitán. Heterogeneous multirobot task allocation for long-endurance missions in dynamic scenarios. *IEEE Trans. Robotics*, 41:6494–6513, 2025.

[9] Ziyang Chen and Zhen Kan. Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications. *Int. J. Robotics Res.*, 44(4):640–664, 2025.

[10] Weiheng Dai, Utkarsh Rai, Jimmy Chiun, Yuhong Cao, and Guillaume Sartoretti. Heterogeneous multi-robot task allocation and scheduling via reinforcement learning. *IEEE Robotics Autom. Lett.*, 10(3):2654–2661, 2025.

[11] Ingy Elsayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, and Lu Feng. Safe multi-agent reinforcement learning via shielding. In *AAMAS*, pages 483–491, 2021.

[12] Francesco Fuggitti and Tathagata Chakraborti. Nl2ltl–a python package for converting natural language (nl) instructions to linear temporal logic (ltl) formulas. In *AAAI*, volume 37, pages 16428–16430, 2023.

[13] Anthony Goeckner, Yueyuan Sui, Nicolas Martinet, Xinliang Li, and Qi Zhu. Graph neural network-based multi-agent reinforcement learning for resilient distributed coordination of multi-robot systems. In *IROS*, pages 5732–5739, 2024.

[14] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[15] Wenjuan Gu, Claudia Archetti, Diego Cattaruzza, Maxime Ogier, Frédéric Semet, and M. Grazia Speranza. Vehicle routing problems with multiple commodities: A survey. *Eur. J. Oper. Res.*, 317(1):1–15, 2024.

[16] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

[18] Jueming Hu, Zhe Xu, Weichang Wang, Guannan Qu, Yutian Pang, and Yongming Liu. Decentralized graph-based multi-agent reinforcement learning using reward machines. *Neurocomputing*, 564:126974, 2024.

[19] Ozgur Kabadurmus and Mehmet S. Erdogan. A green vehicle routing problem with multi-depot, multi-tour, heterogeneous fleet and split deliveries: a mathematical model and heuristic approach. *J. Comb. Optim.*, 45(3):89, 2023.

[20] Qingwen Li, Tang Wai Fan, Lam Sui Kei, and Zhaobin Li. Scalable and energy-efficient task allocation in industry 4.0: Leveraging distributed auction and ibpso. *PLOS ONE*, 20(1): 1–21, 01 2025.

[21] Shunlong Li, Evangelos I Kaisar, Daehan Kwak, Bin Hu, and Dan Liu. Optimizing heterogeneous capacitated vehicle routing with linformer and multi-relationship decoding: A deep reinforcement learning approach. *Transportation Research Record*, 2679(10):39–60, 2025.

[22] Jason Xinyu Liu, Ziyi Yang, Ifrah Idrees, Sam Liang, Benjamin Schornstein, Stefanie Tellex, and Ankit Shah. Grounding complex natural language commands for temporal tasks in unseen environments. In *CoRL*, pages 1084–1110. PMLR, 2023.

[23] Xusheng Luo and Changliu Liu. Simultaneous task allocation and planning for multi-robots under hierarchical temporal logic specifications. *CoRR*, abs/2401.04003, 2024.

[24] Ziyuan Ma, Huajun Gong, Jun Xiong, and Xinhua Wang. Heterogeneous multiagent task allocation based on graph-based convolutional assignment neural network. *IEEE Internet of Things Journal*, 12(11):17281–17299, 2025.

[25] David R. Martin, Brooks A. Butler, Scott Nivison, Magnus Egerstedt, Mohammad Abdullah Al Faruque, and Pramod P. Khargonekar. Collaborative task allocation for heterogeneous multi-robot systems through iterative clustering. *IEEE Robotics Autom. Lett.*, 11(1):33–40, 2026.

[26] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward reshaping. In *ICML*, volume 99, pages 278–287, 1999.

[27] Amir Pnueli. The temporal logic of programs. In *18th annual symposium on foundations of computer science (sfcs 1977)*, pages 46–57. ieee, 1977.

[28] Andoni Rodríguez, Guy Amir, Davide Corsi, César Sánchez, and Guy Katz. Shield synthesis for LTL modulo theories. In *AAAI*, pages 15134–15142, 2025.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, second edition, 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeruIPS*, 30, 2017.

[32] Bruno Salezze Vieira, Glaydston Mattos Ribeiro, and Laura Bahiense. Metaheuristics with variable diversity control and neighborhood search for the heterogeneous site-dependent multi-depot multi-trip periodic vehicle routing problem. *Comput. Oper. Res.*, 153:106189, 2023.

[33] Sihan Wang, Wei Sun, and Min Huang. An adaptive large neighborhood search for the multi-depot dynamic vehicle routing problem with time windows. *Comput. Ind. Eng.*, 191:110122, 2024.

[34] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[35] Jiabing Zhang, Qingxuan Jia, Shiyu Zhang, and Gang Chen. Dynamic and prioritized task scheduling of heterogeneous multi-robot systems using deep reinforcement learning. *Neurocomputing*, 638:130184, 2025.

[36] Bocheng ZHAO, Mingying HUO, Zheng LI, Wenyu FENG, Ze YU, Naiming QI, and Shaohai WANG. Graph-based multi-agent reinforcement learning for collaborative search and tracking of multiple uavs. *Chinese Journal of Aeronautics*, 38 (3), 2025.

[37] Huanfei Zheng and Yue Wang. Parallel decomposition and concurrent satisfaction for heterogeneous multi-robot task and motion planning under temporal logic specifications. *Discret. Event Dyn. Syst.*, 32(2):195–230, 2022.