PACT: PLAN-AUGMENTED COORDINATION VIA TRACE-GRAPHS FOR DYNAMIC MULTI-AGENT COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent approaches have been investigated recently as a means of improving the scalability of reinforcement learning for neural combinatorial optimization. Existing methods assume that all problem instance information is known upfront. However, real-world problems are often *dynamic*: relevant information is progressively revealed as the agents take actions (e.g., arriving at an edge in a routing problem only to discover it is untraversable). In such cases, existing methods fail to coordinate agents to explore and exploit the progressively revealed information. To address the arising coordination challenge, we introduce PACT (**P**lan-**A**ugmented **C**oordination via **T**race-graphs). PACT augments the environment graph with auxiliary nodes representing the plans of agents, which are connected to the resource nodes to indicate future dependencies. The proposed message-passing scheme over the augmented graph grounds inter-agent coordination in the environment and enables proactive, rather than reactive, conflict resolution. We show that PACT improves solution quality in dynamic problems relative to baselines and exhibits the best transfer to larger instances on which, remarkably, it outperforms centralized oracles with privileged access to instance uncertainty.

1 INTRODUCTION

Many routing, scheduling, and allocation problems require several agents to construct a joint combinatorial solution under a shared objective such as total travel cost or makespan, a setting known as multi-agent combinatorial optimization (MACO) (Berto et al., 2025). Existing formulations typically assume that the instance to be solved is fully specified before construction begins. In many real systems, however, relevant information becomes available only through execution: a route may be discovered to be untraversable only when an agent reaches it, or a machine may become known to be unavailable only when an operation is attempted (Pillac et al., 2013; Ouelhadj & Petrovic, 2009). We call this setting **dynamic multi-agent combinatorial optimization** (DMACO).

Classic combinatorial optimization typically assumes fully disclosed instances upfront. Although reactive re-optimization can adapt to newly revealed information, solving must recur continuously (Vieira et al., 2003; Bertsimas et al., 2019), and such reactive paradigms generally do not account for how current decisions affect information and coordination requirements that arise later (Ulmer, 2019). Neural combinatorial optimization (NCO) amortizes the cost of re-solving by learning policies over instance distributions (Kool et al., 2019; Kwon et al., 2020; Kool et al., 2022) and has recently been extended to multi-agent settings (Berto et al., 2025; Luttmann & Xie, 2026). Nonetheless, these NCO approaches still assume fixed and fully observable instances prior to execution.

Since information in DMACO is revealed through agents' actions, we formulate it as a sequential decision-making problem, naturally suited to multi-agent reinforcement learning (MARL) (Bernstein et al., 2002; Oliehoek & Amato, 2016; Yu et al., 2022). Existing MARL approaches typically manage coordination via agent-centric representations (Sunehag et al., 2018; Guestrin et al., 2002; Sukhbaatar et al., 2016). Alongside neural MACO solvers that resolve step-level conflicts during parallel decoding (Berto et al., 2025), these methods broadly share a common limitation: they localize coordination at the agent or step level, without explicitly modeling the shared environmental structure that drives their interdependencies.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In DMACO, coordination requirements are fundamentally induced by the environment. For instance, multiple agents might contend for a capacity-limited resource, or one agent's action might resolve local uncertainty crucial to another's future trajectory. Because these dependencies are grounded in the environment, relying solely on agent-level relations creates a *structural mismatch*: the underlying graph elements driving the coordination are excluded from the representational mechanism. Furthermore, restricting coordination to the current step introduces a *temporal mismatch*. An agent might commit to a trajectory that appears locally conflict-free, only to conflict with another agent's plan several steps later, as illustrated in Figure 1.

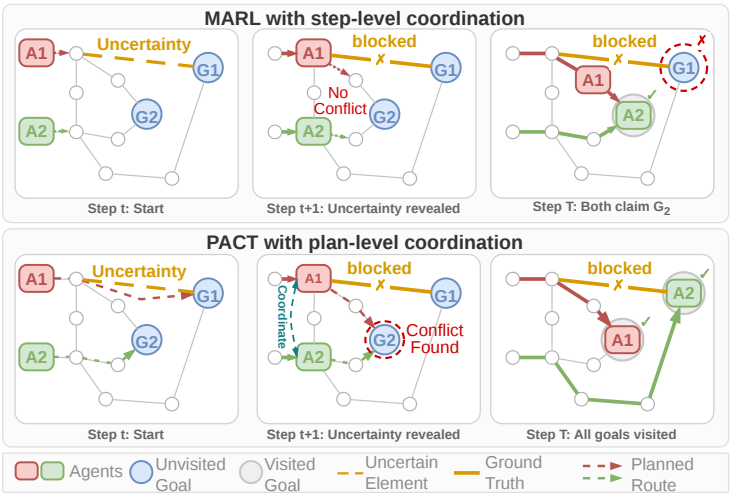


Figure 1: PACT with plan-level coordination prevents goal conflicts exhibited by MARL methods with step-level coordination.

To address these structural and temporal mismatches, we introduce PACT (**P**lan-**A**ugmented **C**oordination via **T**race-graphs), a novel coordination framework for DMACO. First, each agent proposes multiple candidate plans. Every candidate plan is then attached to the environment graph via its projected *trace*, a record of the graph elements on which it places demand. By performing coordination on this plan-augmented trace-graph, PACT enables the underlying environmental structure to explicitly mediate dependencies across agents' future candidate plans. We refer to this mechanism as *environment-mediated coordination*. Our key contributions are as follows:

- We propose PACT, a novel coordination framework for DMACO problems that represents coordination requirements directly through the environmental structure that induces them, rather than through ungrounded relations defined between agents.
- We introduce the plan-augmented trace-graph, which attaches candidate plans to the elements they demand. The proposed message-passing scheme resolves contention over remaining capacities, thereby coordinating agents over future decision steps rather than only the current one.
- On three DMACO problems spanning routing and scheduling, we show that PACT outperforms neural MACO solvers and MARL methods in solution quality and zero-shot generalization on instances up to $64\times$ larger than the training set, at a fraction of the runtime of online replanning.

2 Related Work

Neural Multi-Agent Combinatorial Optimization. NCO amortizes the cost of solving recurring optimization problems by learning solution policies over instance distributions (Bengio et al., 2021; Cappart et al., 2023). Constructive approaches generate solutions autoregressively (Kool et al., 2019; Kwon et al., 2020), with multi-agent extensions coordinating agents' decisions to construct a joint solution. Sequential methods decode agents' actions in turn (Son et al., 2024; Zheng et al., 2024), whereas parallel methods simultaneously decode agents' actions with inter-agent communication (Zong et al., 2022; Hua et al., 2025). These approaches focus on specific MACO problems, while PARCO (Berto et al., 2025) provides a general framework for parallel decoding across MACO problems, resolving conflicting selections through a learned priority-based handler. MACSIM (Luttmann & Xie, 2026) learns through self-improvement to predict joint agent-action assignments that are conflict-free by construction. These methods assume that problem instances are fully specified before solution construction begins and coordinate agents at the current step. They therefore do not explicitly reason about information revealed during execution or dependencies between agents' decisions across future steps.

Coordination in Multi-Agent Reinforcement Learning. Cooperative MARL typically employs centralized training with decentralized execution (Bernstein et al., 2002; Oliehoek & Amato, 2016), under which representative algorithms such as multi-agent PPO (MAPPO) (Yu et al., 2022) train a shared centralized critic, whereas independent PPO (IPPO) (De Witt et al., 2020) uses decentralized critics. Coordination is commonly achieved via value factorization, such as additive decomposition in VDN (Sunehag et al., 2018) and monotonic mixing in QMIX (Rashid et al., 2020), explicit representations like deep coordination graphs (DCG) (Böhmer et al., 2020), or inter-agent communication (Sukhbaatar et al., 2016; Foerster et al., 2016; Das et al., 2019; Hu et al., 2024). In all these families, the coordination medium is agent-centric (e.g., agent utilities, pairwise payoffs, or directed messages). This introduces two limitations when these approaches are applied to DMACO. First, pairwise decompositions struggle to capture simultaneous multi-agent dependencies without incurring the exponential inference costs of higher-order factors (Guestrin et al., 2002; Kok & Vlassis, 2006). Second, these mechanisms coordinate the agents' actions at the current state, whereas conflicts in the future are reflected only implicitly through learned value estimates.

In DMACO settings, PACT addresses both limitations through environment-mediated coordination. By connecting each candidate plan to the graph elements its trace occupies, PACT captures interactions between candidate plans, such as contention for shared capacity or dependence on unrevealed information, at the element that induces them. This element-centric aggregation accommodates multi-agent dependencies without explicit agent-pair factors, while candidate traces expose coordination requirements spanning future decision steps.

3 PRELIMINARIES

We represent a DMACO problem as a graph, where each node is an *element* that a solution selects, assigns, or sequences, and edges encode the relations among elements. We assume that any observation made by one agent is immediately available to the whole team, so that all agents act on the same information. The problem is therefore modeled as a fully cooperative multi-agent partially observable Markov decision process (MPOMDP) (Oliehoek & Amato, 2016), defined by the tuple $\langle \mathcal{M}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{M}}, \mathcal{T}, O, C, \gamma \rangle$, where $\mathcal{M}$ is the team of agents, $\mathcal{S}$ the states, $\{\mathcal{A}_i\}_{i\in\mathcal{M}}$ the action space for each agent, $\mathcal{T}$ the transition function, O the observation function, C the cost function, and $\gamma \in (0, 1]$ the discount factor. We factorize the state space $\mathcal{S}$ into observable graph states $\mathcal{G}_t$ and an unobserved latent realization ξ, with $O(\mathcal{G}_t, \xi) = \mathcal{G}_t$. At the start of each episode we sample $\xi \sim p(\xi)$, and it is held fixed for the episode. The observed graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, X_t)$ consists of the elements $\mathcal{V}_t$, the relations $\mathcal{E}_t$ among them, and the attributes X_t of each element and each agent. The attributes X_t also determine the set $\mathcal{M}_t \subseteq \mathcal{M}$ of agents active at decision step t. Each active agent $i \in \mathcal{M}_t$ selects an action $a_{i,t} \in \mathcal{A}_i(\mathcal{G}_t)$ from its own policy π_i conditioned on the graph observation $\mathcal{G}_t$, where $\mathcal{A}_i(\mathcal{G}_t) \subseteq \mathcal{V}_t \cup \{\varnothing\}$, so that agent i commits either to a single element or to none. We write $\pi = \{\pi_i\}_{i\in\mathcal{M}}$ for the team policy. The state then advances according to $\mathcal{G}_{t+1} \sim \mathcal{T}(\cdot \mid \mathcal{G}_t, \mathbf{a}_t, \xi)$, where the joint action $\mathbf{a}_t = \{a_{i,t}\}_{i\in\mathcal{M}_t}$. Each decision step incurs a cost $C(\mathcal{G}_t, \mathbf{a}_t, \xi) \in \mathbb{R}_{\geq 0}$, and the agents minimize the expected discounted cost $J(\pi) = \mathbb{E}_{\xi,\pi}\Big[\sum_t \gamma^t\, C(\mathcal{G}_t, \mathbf{a}_t, \xi)\Big]$.

Capacities and demands serve as the general interface for defining resource constraints in combinatorial optimization (Blazewicz et al., 1983) and constraint programming (Aggoun & Beldiceanu, 1993). We assign a capacity $\kappa_t(v) \geq 0$ to each element $v \in \mathcal{V}_t$, recorded in the attributes X_t. The action a_i from agent i places a demand $d(v, a_{i,t}) \geq 0$ on the element v. A unit capacity $\kappa(v) = 1$ makes an element exclusive, which covers a goal location claimed by a single traveler (Toth & Vigo, 2014) and a job processed by a single machine (Pinedo, 1995), while larger capacities describe resources that several agents may hold simultaneously. At decision step t, the demand on element v from all active agents is $d_t(v) = \sum_{i\in\mathcal{M}_t} d(v, a_{i,t})$. The joint actions permitted by the remaining capacities form the set

$$\mathcal{A}(\mathcal{G}_t) = \left\{ a = \{a_i\}_{i\in\mathcal{M}_t} \in \prod_{i\in\mathcal{M}_t} \mathcal{A}_i(\mathcal{G}_t) \;:\; d_t(v) \leq \kappa_t(v) \text{ for every } v \in \mathcal{V}_t \right\}. \tag{1}$$

Eq. (1) is the only feasibility constraint that couples the agents, and it is a conjunction of conditions taken element by element.

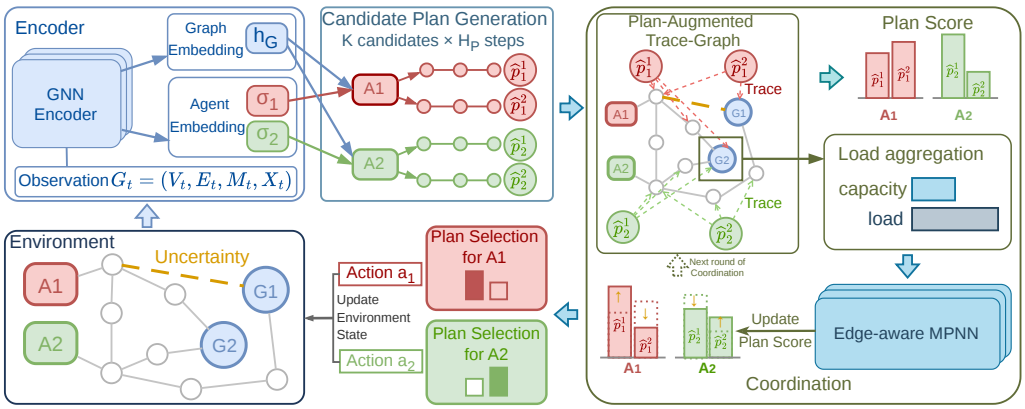


Figure 2: Overview of PACT. After encoding graph observations, agents independently generate candidate plans to form a plan-augmented trace-graph for multi-step coordination. The joint action $\mathbf{a}_t$ comprises the initial step of each selected plan.

4 Methodology

We now present PACT, whose overall architecture is illustrated in Figure 2. Each agent first generates a set of candidate plans instead of committing to a single next action (Section 4.1). The candidate plans are incorporated into the graph through the elements they occupy, forming a plan-augmented trace-graph over which inter-agent coordination is performed (Section 4.2). Each agent subsequently selects one of its coordinated candidate plans and executes its first action (Section 4.3). Finally, all modules are trained end to end (Section 4.4). The complete forward pass of PACT is summarized in Algorithm 1.

4.1 Candidate Plan Generation

As was illustrated in Figure 1, coordination over individual actions is inherently myopic: agents whose current actions are compatible may still contend for the same resources several steps ahead. PACT therefore coordinates over short-horizon candidate plans. However, planning in the joint action space scales exponentially with both the number of agents and the planning horizon (Bernstein et al., 2002; Kearns et al., 2002). We instead decouple plan proposal from multi-agent coordination: each agent independently generates a compact set of candidate plans, while interactions among these plans are handled by the coordination mechanism in Section 4.2.

Given the current graph observation $\mathcal{G}_t$, a GNN-based encoder Enc_θ, whose parameters θ are shared with every other learned module of PACT, extracts graph-related representations h_G together with an agent-related representation σ_i for each active agent,

$$\big(h_G,\ \{\sigma_i\}_{i\in\mathcal{M}_t}\big) = \mathrm{Enc}_\theta(\mathcal{G}_t). \tag{2}$$

The graph representation h_G describes the decision-relevant structure and elements of the current problem instance, while σ_i provides the agent-specific context needed to specialize the graph representation to agent i.

For each agent i, PACT generates K candidate plans, each consisting of a sequence of H_p actions,

$$\hat{p}_i^k = \big(\hat{a}_{i,t}^k,\ \hat{a}_{i,t+1}^k,\ \ldots,\ \hat{a}_{i,t+H_p-1}^k\big), \qquad k = 1,\ldots,K. \tag{3}$$

Each candidate plan is constructed through a rollout governed by a learned candidate plan generation policy $\pi_\theta^{\mathrm{plan}}$. To ensure diversity among the K candidate plans, their first actions are drawn jointly without replacement from the policy at the initial rollout state $\pi_\theta^{\mathrm{plan}}(\mathcal{G}_t)$. Following this initialization, each of the K plans unrolls independently. At each rollout step j, $\pi_\theta^{\mathrm{plan}}$ conditions on the encoded graph h_G, the agent representation σ_i, and the rollout state at step j, $\hat{\mathcal{G}}_{t+j,i}^k$ to extend the partial plan, ensuring that subsequent decisions dynamically reflect the accumulated consequences of preceding actions.

4.2 PLAN-AUGMENTED TRACE-GRAPH AND COORDINATION

Recall the routing scenario shown in Figure 1. In this scenario, two agents move apart at the current step but then both claim the same goal several steps later. Their current actions place demand on different elements, so Eq. (1) holds at every step until they arrive, and the demand that must be reconciled is located at the goal, where the final two actions together exceed its remaining capacity $\kappa_t(v)$. PACT therefore records, for each candidate plan, which elements it would occupy, the induced demand, and the step of the plan. We refer to this record as the candidate plan's *trace*,

$$\mathrm{Tr}(\hat{p}_i^k) \;=\; \big\{\, \big(v,\; d(v, \hat{a}_{i,t+j}^k),\; j\big) \;:\; d(v, \hat{a}_{i,t+j}^k) > 0,\;\; j = 0, \ldots, H_p - 1 \,\big\}, \tag{4}$$

where j indexes the step of the plan at which the demand is placed.

Once K candidate plans are generated in parallel, their traces form the *plan-augmented trace-graph* $\chi_t \;=\; \big(\mathcal{V}_t \cup \hat{\mathcal{V}}_t,\;\; \mathcal{E}_t \cup \hat{\mathcal{E}}_t\big)$, where candidate plan nodes $\hat{\mathcal{V}}_t = \{\hat{p}_i^k : i \in \mathcal{M}_t,\, k \le K\}$ and trace arcs $\hat{\mathcal{E}}_t = \{(\hat{p}, v) : (v, \cdot, \cdot) \in \mathrm{Tr}(\hat{p})\}$. Each arc carries the demand and the plan step that produced it. Arcs join candidate plans to elements but not to one another. This forces information exchange between agents to occur through the graph structure. Thus, the same graph structure captures both the original combinatorial problem and the prospective interactions introduced by the plans.

PACT coordinates the candidate plans by iteratively alternating between two operations in order to compute a set of *preferences*. First, the resource contention induced by the agents' current plan preferences is estimated. Second, preferences are updated through message passing on the plan-augmented trace-graph χ_t. Concretely, let $s_i^{(r)}(k)$ denote the *score* of plan k of agent i after coordination round $r = 0, \ldots, R$. The corresponding *preference* is:

$$q_i^{(r)}(k) = \frac{\exp(s_i^{(r)}(k))}{\sum_{k'=1}^{K} \exp(s_i^{(r)}(k'))}. \tag{5}$$

Every node of χ_t carries a representation $h^{(r)}$ at round r, whose initial value $h^{(0)}$ is given by h_G for element nodes and by an encoding of the node's trace for candidate plan nodes. A scoring head f_{score} turns a candidate plan's representation into its score, so that $s_i^{(0)}(k) \;=\; \eta\, \tanh\big(f_{\mathrm{score}}(h^{(0)}(\hat{p}_i^k))\big)$ before information is exchanged between agents, where $\eta > 0$ is a fixed step size bound.

Given the current preferences $q_i^{(r)}$, each element aggregates the demand induced by all candidate plans that reach it. We define $d(v, \hat{p}_i^k) = \sum_j d(v, \hat{a}_{i,t+j}^k)$ as the total demand that $\hat{p}_i^k$ places on v across its steps. The load on element v at round r is the preference-weighted demand of all plans,

$$L^{(r)}(v) = \sum_{i \in \mathcal{M}_t} \sum_{k=1}^{K} q_i^{(r)}(k) \;\times d(v, \hat{p}_i^k). \tag{6}$$

Hence, $L^{(r)}(v)$ measures the expected demand induced by the agents' current preferences $q_i^{(r)}$ over their candidate plans. The load is compared with the remaining capacity $\kappa_t(v) - d_t(v)$ of each element and injected into its representation through a learned projection W_{load}, normalized by the capacity $\kappa_t(v)$,

$$\tilde{h}^{(r)}(v) = h^{(r)}(v) + W_{\mathrm{load}} \left[\frac{L^{(r)}(v)}{\kappa_t(v)},\; \frac{\mathrm{ReLU}\big(L^{(r)}(v) - (\kappa_t(v) - d_t(v))\big)}{\kappa_t(v)} \right], \tag{7}$$

where $\frac{L^{(r)}(v)}{\kappa_t(v)}$ describes resource utilization, while $\big(L^{(r)}(v) - (\kappa_t(v) - d_t(v))\big)$ explicitly identifies demand beyond the remaining capacity. Normalizing by capacity provides a common scale for resources with different capacities.

An edge-aware message-passing layer, shared across rounds, then propagates this information over the plan-augmented trace-graph,

$$h^{(r+1)} = \mathrm{MPNN}\big(\tilde{h}^{(r)}, \chi_t\big), \tag{8}$$

$$s_i^{(r+1)}(k) = s_i^{(r)}(k) + \eta\, \tanh\big(f_{\mathrm{score}}(h^{(r+1)}(\hat{p}_i^k))\big). \tag{9}$$

Algorithm 1 PACT forward pass at decision step t

Input: graph observation $\mathcal{G}_t$
Encode $\mathcal{G}_t$ once into h_G and $\{\sigma_i\}$ (Eq. (2))
for each agent $i \in \mathcal{M}_t$ (in parallel) **do**
  Generate K candidate plans $\hat{p}_i^1, \ldots, \hat{p}_i^K$ through rollout, with first actions drawn jointly without replacement (Section 4.1)
end for
Construct the plan-augmented trace-graph χ_t from the candidate plan traces (Eq. (4))
Initialize candidate plan scores $s_i^{(0)}$ and preferences $q_i^{(0)} = \mathrm{softmax}(s_i^{(0)})$ (Eq. (5))
for coordination round $r = 0, \ldots, R-1$ **do**
  Compute the load $L^{(r)}(v)$ induced by the current plan preferences (Eq. (6))
  Inject capacity-aware load features and perform message passing over χ_t (Eqs. (7) and (8))
  Update candidate plan scores $s_i^{(r+1)}$ and preferences $q_i^{(r+1)}$ (Section 4.2) (Eq. (9))
end for
for each agent $i \in \mathcal{M}_t$ (in parallel) **do**
  Sample $z_i \sim \mathrm{softmax}(s_i^{(R)}/\tau)$ and set $a_{i,t} = \hat{p}_i^{z_i}[0]$ (Section 4.3)
end for
Output: Joint action $\mathbf{a}_t = \{a_{i,t}\}_{i \in \mathcal{M}_t}$

The updated scores are renormalized over the K candidate plans of each agent to obtain $q_i^{(r+1)}$, which in turn induces a new load profile through Eq. (6). The candidate plan scores are therefore updated iteratively through coordination, allowing an agent's preference over its own plans to respond to the evolving preferences of the rest of the team.

4.3 Plan Selection and Execution

After R coordination rounds, each agent selects a candidate plan based on its final coordinated scores, $z_i \sim \mathrm{softmax}(s_i^{(R)}/\tau)$, where τ is a learned temperature controlling the sharpness of selection over candidate plans. Each agent then executes the first action of its chosen plan, $a_{i,t} = \hat{p}_i^{z_i}[0]$. The resulting joint action $\mathbf{a}_t = \{a_{i,t}\}_{i \in \mathcal{M}_t}$ is applied to the environment via $\mathcal{T}$, which incurs the cost $C(\mathcal{G}_t, \mathbf{a}_t, \xi)$, reveals the components of ξ associated with $\mathbf{a}_t$, and yields the subsequent graph observation $\mathcal{G}_{t+1}$. Following the receding-horizon principle, PACT regenerates and coordinates new candidate plans from $\mathcal{G}_{t+1}$ at each step. Consequently, while future actions within a candidate plan serve as look-ahead information for multi-agent coordination, execution remains closed-loop and adapts to newly revealed information.

4.4 Policy Training

PACT learns its plan generation and plan-selection policies from the return. Unlike a conventional policy in which the sampled policy output is the environment action, PACT produces each agent's action through a structured sequence of stochastic decisions: candidate plans are first generated and subsequently selected after coordination. The policy objective therefore needs to account for the internal decisions that give rise to the executed action.

We decompose the policy update into three components: *candidate plan proposal* (prop), *plan rollout* (roll), and *plan selection* (sel). For each component $g \in \{\mathrm{prop}, \mathrm{roll}, \mathrm{sel}\}$, we optimize the clipped surrogate objective:

$$\mathcal{L}_g(\theta) = \mathbb{E}_t \left[\sum_{i \in \mathcal{M}_t} \min \left(\rho_{i,t}^g(\theta) \hat{A}_t,\ \mathrm{clip}\left(\rho_{i,t}^g(\theta), 1-\epsilon_g, 1+\epsilon_g\right) \hat{A}_t \right) \right], \tag{10}$$

where $\rho_{i,t}^g$ is the importance ratio between the probabilities that the current and the behavior policies assign to agent i's decisions in component g. For prop, these decisions are the K first actions, and the ratio compares their joint probability as an ordered set drawn without replacement. For roll, they are the actions at steps $j \geq 1$ of the selected plan $\hat{p}_i^{z_i}$. For sel, the decision is the selected index z_i. The advantage $\hat{A}_t$ is estimated with generalized advantage estimation (Schulman et al., 2015)

from a critic conditioned on the graph observation $\mathcal{G}_t$, and ϵ_g is the clipping width. The overall objective sums the component objectives as $\mathcal{L}_{\mathrm{PACT}}(\theta) = \mathcal{L}_{\mathrm{prop}}(\theta) + \mathcal{L}_{\mathrm{roll}}(\theta) + \mathcal{L}_{\mathrm{sel}}(\theta)$. Clipping each component separately preserves PACT's structured hierarchy while allowing a single return signal to jointly train proposal, rollout, and selection.

5 Experiments

5.1 Experimental Setting

Problems. We evaluate PACT on three representative DMACO problems, spanning both routing and scheduling domains. In the **Multi-Agent Canadian Traveller Problem (MACTP)** (Papadimitriou & Yannakakis, 1991; Shiri & Salman, 2017), a team traverses a graph to collectively visit goal nodes at minimum total travel cost, with edge traversability revealed upon reaching an endpoint or attempting a crossing. Travelers are agents and goal nodes have unit capacity. In the **Dynamic Flexible Job-Shop Problem (DFJSP)** (Ouelhadj & Petrovic, 2009), jobs comprise precedence-constrained operations assigned to eligible machines; whether an uncertain machine is operational or broken is revealed when an operation is dispatched to it. In the **Non-Clairvoyant Flexible Flow Shop Problem (NCFFSP)** (Motwani et al., 1994), jobs pass through ordered stages of interchangeable machines whose latent processing-time factors are revealed upon dispatch. Both scheduling problems minimize makespan, with machines acting as agents and jobs as unit-capacity elements. Appendix A provides the formal models and revelation rules.

Baselines. We compare PACT with two multi-agent NCO methods, PARCO and MACSIM, which are evaluated as centralized joint-construction solvers through task wrappers, retaining their original training objectives and model configurations. For PACT, PARCO, and MACSIM two decoding strategies named *greedy (g.)* and *sampling (s.)* are used. We also compare four cooperative MARL baselines (MAPPO, IPPO, VDN, and DCG), using matched graph encoders over the common observation interface and shared parameters across agents. MAPPO, IPPO, and VDN produce per-agent actions, whereas DCG coordinates joint actions through max-plus message passing on an agent coordination graph. PACT uses a critic conditioned on the graph observation and performs decentralized execution with communication through the plan-augmented trace-graph, in which each agent selects its own candidate plan. Gurobi (Gurobi Optimization, LLC, 2026) and OR-Tools (Perron & Furnon) provide online replanning baselines and oracle references. Appendix B.1 details the baseline implementations and adaptations.

Training. For each problem, all learning-based methods are trained on the same instance set with a mixture of small sizes. The mixture spans $n \in \{10, 13, 16\}$ nodes for MACTP, $N \in \{12, 16, 20\}$ jobs for DFJSP, and $N \in \{12, 16, 20\}$ jobs with 3 stages and 3 or 4 machines per stage for NCFFSP. Dataset generation seeds are fixed, and each method is trained with 10 independent seeds. The training configurations are given in Appendix B.

Evaluation. All methods use the same 100 test instances per configuration, and learning-based results are averaged across the 10 training seeds. Instance sizes are denoted by nodes $\times$ agents $\times$ goals ($n \times A \times G$) for MACTP, jobs $\times$ machines $\times$ operations ($N \times M \times O$) for DFJSP, and jobs $\times$ machines per stage $\times$ stages ($N \times M_\ell \times S$) for NCFFSP. The first configuration in each problem block is in-distribution (ID), matching the training mixture's largest size; the remaining four evaluate zero-shot transfer (ZST) to larger instances. The final two NCFFSP configurations also increase the stage count from 3 to 4. Appendix C.2 details the evaluation protocols.

5.2 Main Results

Table 1 shows that PACT, using the same framework across all three problems with decentralized execution and coordination through the plan-augmented trace-graph, attains the lowest cost among learning-based methods in almost every configuration under greedy decoding and in every configuration under sampling. Each baseline instead degrades in some regime: the MARL baselines exceed GPU memory on large MACTP instances, PARCO's success collapses on the largest MACTP and DFJSP instances, and VDN, DCG, and MACSIM leave a large fraction of DFJSP episodes incomplete under greedy decoding. The advantage of PACT grows under zero-shot transfer and is most pronounced on the largest MACTP configuration, where it reduces greedy cost by 34% relative to MACSIM while maintaining 99.6% success. Compared with online replanning, PACT reaches

Table 1: Results on three DMACO problems under ID and ZST evaluation. Cost ↓, Time (s) ↓, and Success (%) ↑. N/A marks unsupported configurations, T/O marks runs without a feasible solution within the time limit, and OOM marks runs that exceed GPU memory.

MACTP

$n \times A \times G$		16×4×6 (ID)			32×6×10 (ZST)			128×12×24 (ZST)			512×24×48 (ZST)			1024×32×64 (ZST)		
	Metric	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time
Oracle	*Gurobi (10m)*	*2.84*	*100.0*	*0.18*	*4.28*	*100.0*	*0.25*	*9.52*	*100.0*	*386.5*	*21.9*	*100.0*	*627.2*	*3132.5*	*93.0*	*652.3*
	OR-Tools (10m)	*2.84*	*100.0*	*0.66*	*4.28*	*100.0*	*37.5*	*13.0*	*100.0*	*624.7*	*917.7*	*97.0*	*633.8*	*T/O*	*T/O*	*T/O*
Replan	Gurobi	5.02	90.0	0.42	10.1	79.0	1.06	95.0	37.0	13.9	T/O	T/O	T/O	T/O	T/O	T/O
	OR-Tools	4.08	93.0	0.92	23.5	34.0	6.37	231.9	39.0	121.3	T/O	T/O	T/O	T/O	T/O	T/O
Greedy	MAPPO (*g.*)	3.83	96.8	0.027	6.78	98.1	0.020	32.8	95.7	0.79	OOM	OOM	OOM	OOM	OOM	OOM
	IPPO (*g.*)	3.92	97.1	0.015	6.84	98.8	0.033	28.6	97.2	0.63	OOM	OOM	OOM	OOM	OOM	OOM
	VDN (*g.*)	**3.52**	98.9	0.012	5.84	99.6	0.019	20.7	99.9	0.38	OOM	OOM	OOM	OOM	OOM	OOM
	DCG (*g.*)	3.68	99.3	0.011	8.44	98.1	0.048	70.0	93.3	0.99	OOM	OOM	OOM	OOM	OOM	OOM
	PARCO (*g.*)	4.56	98.9	0.004	10.8	95.0	0.017	195.1	41.2	0.11	1722.0	4.2	0.35	3999.2	4.5	0.73
	MACSIM (*g.*)	7.36	89.9	0.038	12.7	95.2	0.035	34.8	99.6	0.047	90.2	100.0	0.18	131.3	100.0	2.20
	PACT (*g.*)	3.66	99.1	0.056	**5.79**	99.6	0.064	**18.6**	99.7	0.34	**72.3**	99.0	2.61	**86.1**	99.6	11.0
Sampling	PARCO (*s.*)	3.35	100.0	0.094	5.50	100.0	0.15	17.8	99.7	2.67	372.6	86.3	9.83	1341.6	71.3	27.5
	MACSIM (*s.*)	3.27	100.0	0.48	6.32	100.0	0.68	24.0	100.0	1.91	84.2	100.0	42.0	125.8	100.0	381.7
	PACT (*s.*)	**2.89**	100.0	5.27	**4.66**	100.0	5.45	**15.5**	100.0	10.5	**50.3**	100.0	51.9	**84.0**	100.0	136.9

DFJSP

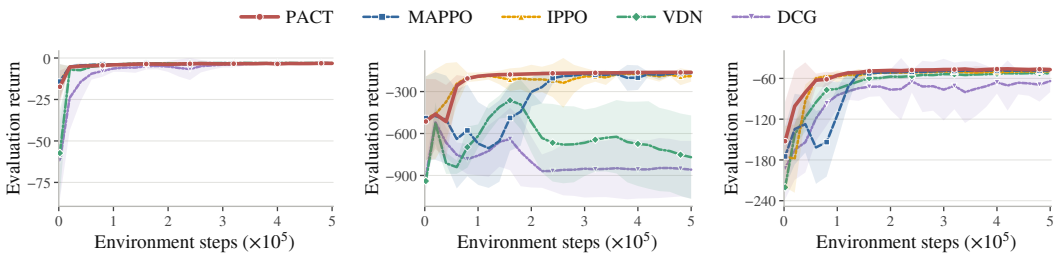
$N \times M \times O$		20×8×6 (ID)			30×14×6 (ZST)			50×16×8 (ZST)			80×20×8 (ZST)			120×24×10 (ZST)		
	Metric	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time
Oracle	*Gurobi (10m)*	*154.9*	*100.0*	*624.6*	*133.7*	*100.0*	*626.3*	*291.7*	*100.0*	*614.2*	*T/O*	*T/O*	*T/O*	*T/O*	*T/O*	*T/O*
	OR-Tools (10m)	*148.3*	*100.0*	*624.2*	*126.1*	*100.0*	*624.2*	*221.6*	*100.0*	*600.6*	*T/O*	*T/O*	*T/O*	*T/O*	*T/O*	*T/O*
Replan	Gurobi	**156.8**	100.0	28.8	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O
	OR-Tools	172.7	87.0	61.6	149.4	100.0	76.3	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O	T/O
Greedy	MAPPO (*g.*)	170.2	100.0	0.022	148.4	100.0	0.021	245.1	100.0	0.070	299.0	100.0	0.24	429.2	100.0	0.84
	IPPO (*g.*)	207.7	96.1	0.051	156.1	99.7	0.054	295.6	98.7	0.24	380.1	98.6	0.87	584.4	98.5	3.45
	VDN (*g.*)	632.1	54.4	0.056	688.7	66.5	0.076	1641.3	57.7	0.33	2498.4	59.6	1.27	4311.0	59.9	5.54
	DCG (*g.*)	986.6	15.1	0.088	1476.8	13.4	0.13	3089.9	7.4	0.61	5017.7	5.4	2.72	9296.8	0.5	13.1
	PARCO (*g.*)	170.5	100.0	0.004	146.0	100.0	0.004	258.4	99.7	0.015	336.2	97.4	0.054	2578.1	7.6	0.35
	MACSIM (*g.*)	608.3	54.7	0.18	596.9	70.1	0.21	1241.3	60.2	0.48	1305.2	58.0	0.84	1866.3	61.2	1.64
	PACT (*g.*)	162.5	100.0	0.093	**141.5**	100.0	0.10	**233.2**	100.0	0.25	**283.9**	100.0	0.77	**398.4**	100.0	2.41
Sampling	PARCO (*s.*)	165.0	100.0	0.087	142.8	100.0	0.096	249.7	100.0	0.18	314.5	100.0	0.28	490.3	100.0	2.30
	MACSIM (*s.*)	156.2	100.0	0.20	136.4	100.0	0.25	242.3	100.0	0.93	307.2	100.0	1.83	458.3	100.0	5.24
	PACT (*s.*)	**155.9**	100.0	10.5	**134.1**	100.0	10.5	**232.4**	100.0	23.6	**279.6**	100.0	54.4	**396.5**	100.0	112.1

NCFFSP

$N \times M_\ell \times S$		20×4×3 (ID)			30×6×3 (ZST)			50×7×3 (ZST)			80×6×4 (ZST)			120×8×4 (ZST)		
	Metric	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time	Cost	Succ.	Time
Oracle	*Gurobi (10m)*	*34.3*	*100.0*	*624.5*	*30.7*	*100.0*	*600.2*	*41.2*	*100.0*	*601.6*	*91.2*	*100.0*	*602.2*	*183.5*	*100.0*	*607.3*
	OR-Tools (10m)	*32.0*	*100.0*	*124.0*	*26.9*	*100.0*	*603.6*	*36.0*	*100.0*	*600.2*	*67.5*	*100.0*	*600.3*	*70.8*	*100.0*	*600.2*
Replan	Gurobi	37.5	100.0	15.7	**34.1**	100.0	19.3	225.1	55.0	31.4	T/O	T/O	T/O	T/O	T/O	T/O
	OR-Tools	**37.1**	100.0	6.62	36.5	100.0	13.6	53.6	100.0	32.2	351.3	96.0	112.5	T/O	T/O	T/O
Greedy	MAPPO (*g.*)	46.5	97.4	0.019	38.0	99.2	0.020	49.4	99.3	0.065	94.5	98.2	0.33	93.8	98.4	1.13
	IPPO (*g.*)	47.3	98.5	0.017	39.3	99.3	0.020	52.6	98.9	0.070	99.8	97.8	0.30	97.1	98.6	1.05
	VDN (*g.*)	50.0	97.8	0.017	44.2	96.8	0.027	63.9	93.2	0.12	151.0	84.4	1.04	186.1	81.6	4.62
	DCG (*g.*)	84.1	71.7	0.036	102.4	58.9	0.069	152.4	59.0	0.26	376.7	50.3	1.64	513.9	48.4	6.98
	PARCO (*g.*)	42.9	99.3	0.005	49.8	98.8	0.009	66.8	98.8	0.010	N/A	N/A	N/A	N/A	N/A	N/A
	MACSIM (*g.*)	64.7	82.9	0.067	55.4	91.7	0.057	98.4	86.6	0.11	212.6	82.7	0.29	213.9	89.1	0.35
	PACT (*g.*)	41.5	100.0	0.049	35.9	100.0	0.063	**48.8**	99.9	0.17	**83.4**	99.7	0.51	**86.7**	99.4	1.84
Sampling	PARCO (*s.*)	37.3	100.0	0.085	39.6	100.0	0.099	57.4	100.0	0.13	N/A	N/A	N/A	N/A	N/A	N/A
	MACSIM (*s.*)	39.0	100.0	0.21	36.1	100.0	0.32	50.7	100.0	0.71	97.3	100.0	2.09	109.6	100.0	3.87
	PACT (*s.*)	**36.2**	100.0	5.35	**35.4**	100.0	6.17	**44.8**	100.0	11.7	**82.5**	100.0	32.5	**85.6**	100.0	97.0

comparable or lower cost with sampling and runs at least seven times faster under greedy decoding wherever replanning completes, whereas replanning loses success or exceeds its time limit as instances grow. We note that, while *Oracle* methods are not directly comparable due to having privileged access to the uncertainty ξ, they nevertheless may not yield optimal solutions on the largest instances, since they are limited by the time budget. We provide a breakdown of PACT's end-to-end wall clock runtime into environment stepping, encoding, candidate plan generation, coordination, plan selection, and other costs in Appendix C.3.

Validation returns obtained during training are reported in Figure 3. On MACTP, PACT and the MARL baselines reach similar final returns. On DFJSP, PACT reaches high-return performance earlier than MAPPO and maintains it thereafter, with lower variability across seeds than VDN and DCG, whose returns deteriorate after initial gains. On NCFFSP, PACT also learns faster than MAPPO and attains a higher final mean return. We present PARCO and MACSIM's training curves



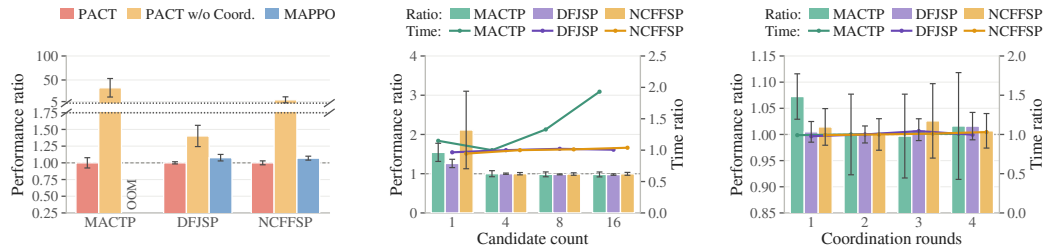
(a) MACTP (b) DFJSP (c) NCFFSP

Figure 3: Validation return during training. Curves show the mean across ten seeds; shaded regions indicate one standard deviation. Higher is better.

separately in Appendix C.1 because they require substantially more training steps. Despite larger budgets, PARCO attains lower best mean returns than PACT on all three problems, while MACSIM converges to a comparable return on MACTP and DFJSP but remains lower on NCFFSP.

5.3 Ablation Study

We ablate PACT on the largest zero-shot transfer configuration of each problem under greedy evaluation and report cost and runtime relative to the reference model with $K = 4$ candidate plans and $R = 2$ coordination rounds (Figure 4). Appendix C.4 details the protocol and the analysis of each component. Coordination is the main source of PACT's gains. Removing message passing on the plan-augmented trace-graph degrades performance to below that of MAPPO on DFJSP and NCFFSP, so candidate generation alone does not explain the improvement. Coordination also depends on having alternatives to choose from. With a single candidate plan per agent, performance degrades on every problem, whereas four candidates provide most of the benefit and larger sets yield no consistent gain. A second coordination round matters on MACTP, where plans span $H_p = 4$ steps and capture multi-step interactions, while one round suffices on the scheduling problems with $H_p = 1$. Further rounds bring no consistent improvement at nearly unchanged runtime.



(a) Effect of Coordination (b) Number of Candidate Plans (c) Coordination Rounds

Figure 4: Analysis of PACT components.

6 Conclusion

In this paper, we introduced PACT, a framework for DMACO that enables agents to coordinate over candidate plans through the environmental structure that induces their dependencies. By augmenting the environment graph with candidate plan traces and performing capacity-aware message passing over the resulting trace-graph, PACT extends coordination beyond the current decision step while retaining closed-loop execution as new information is revealed. Our experiments on routing and scheduling problems demonstrate that PACT consistently achieves strong solution quality and completion success across both in-distribution and zero-shot transfer settings, with particularly strong generalization to larger instances. In terms of limitations, the global communication assumption behind MPOMDPs may be relaxed in future work by imposing communication radiuses inside which information is shared. Additionally, PACT samples independently from agents' policies π_i to construct the joint action at inference time. Stronger joint actions may be obtained by reasoning over the policies of other agents, for example by maintaining and updating beliefs.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, and to implement methods. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, or interpret results. Cleaning and reformatting datasets and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to draft parts of the research paper, discover research topics or identify gaps, brainstorm, source/search for information, edit the research paper to improve readability, identify relevant literature, and assist with translation. We have reviewed all AI-assisted work. We checked all sources and LLM-generated research ideas for potential plagiarism through a manual literature review. All LLM-generated code was validated and tested for correctness. All LLM-generated writing has been manually reviewed for accuracy. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Source code used to implement the PACT method will be released upon publication of the paper. Appendix A gives the formal model and revelation rules of each problem, and Appendix B.3 specifies the instance generators. Appendix B describes the baseline implementations, and Appendix B.4 and Appendix B.5 report the training configurations and the compute environment. Appendix D details the algorithms of PACT.

REFERENCES

Abderrahmane Aggoun and Nicolas Beldiceanu. Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and computer modelling*, 17(7):57–73, 1993.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Federico Berto, Chuanbo Hua, Laurin Luttmann, Jiwoo Son, Junyoung Park, Kyuree Ahn, Changhyun Kwon, Lin Xie, and Jinkyoo Park. PARCO: Parallel autoregressive models for multi-agent combinatorial optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=9F2Cmgo17M.

Dimitris Bertsimas, Patrick Jaillet, and Sébastien Martin. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research*, 67(1):143–162, 2019.

Matteo Bettini, Amanda Prorok, and Vincent Moens. BenchMARL: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024.

J. Blazewicz, J.K. Lenstra, and A.H.G.Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(83)90012-4. URL https://www.sciencedirect.com/science/article/pii/0166218X83900124.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pp. 980–991. PMLR, 2020.

Quentin Cappart, Didier Chételat, Elias B Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *International Conference on machine learning*, pp. 1538–1546. PMLR, 2019.

Christian Schroeder De Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the StarCraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, volume 29, 2016.

Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *International Conference on Machine Learning*, pp. 227–234, 2002.

Gurobi Optimization, LLC. Gurobi optimizer reference manual. `https://www.gurobi.com`, 2026.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *International Conference on Learning Representations*, volume 2024, pp. 12963–12978, 2024.

Chuanbo Hua, Federico Berto, Jiwoo Son, Seunghyun Kang, Changhyun Kwon, and Jinkyoo Park. CAMP: Collaborative Attention Model with Profiles for Vehicle Routing Problems. In *Proceedings of the 2025 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2025. `https://github.com/ai4co/camp`.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2):193–208, 2002.

Jelle R. Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7(65):1789–1828, 2006. URL `http://jmlr.org/papers/v7/kok06a.html`.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByxBFsRqYm`.

Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. In *International conference on integration of constraint programming, artificial intelligence, and operations research*, pp. 190–213. Springer, 2022.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in neural information processing systems*, volume 33, pp. 21188–21198, 2020.

Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5138–5149, 2021.

Laurin Luttmann and Lin Xie. Multi-action self-improvement for neural combinatorial optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=6KrETIaOYD`.

Julian Minder, Florian Grötschla, Joël Mathys, and Roger Wattenhofer. SALSA-CLRS: A sparse and scalable benchmark for algorithmic reasoning. In *The Second Learning on Graphs Conference*, 2023. URL `https://openreview.net/forum?id=PRapGjDGFQ`.

Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(94)90151-1. URL `https://www.sciencedirect.com/science/article/pii/0304397594901511`.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

Djamila Ouelhadj and Sanja Petrovic. A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling*, 12(4):417–431, 2009.

Cemal Özgüven, Lale Özbakır, and Yasemin Yavuz. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied mathematical modelling*, 34(6):1539–1548, 2010.

Christos H Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

Laurent Perron and Vincent Furnon. OR-Tools. URL `https://developers.google.com/optimization/`.

Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

Michael. Pinedo. *Scheduling : theory, algorithms, and systems*. Prentice-Hall international series in industrial and systems engineering. Prentice Hall, Englewood Cliffs, N.J, 1995. ISBN 0137067577.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020. URL `http://jmlr.org/papers/v21/20-081.html`.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Davood Shiri and F. Sibel Salman. On the online multi-agent O-D k-Canadian traveler problem. *Journal of Combinatorial Optimization*, 34(2):453–461, 2017.

Jiwoo Son, Minsu Kim, Sanghyeok Choi, Hyeonah Kim, and Jinkyoo Park. Equity-transformer: Solving NP-hard min-max routing problems as sequential generation with equity context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20265–20273, 2024.

Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2022.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in neural information processing systems*, volume 29, 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

Marlin W Ulmer. Anticipation versus reactive reoptimization for dynamic vehicle routing with stochastic requests. *Networks*, 73(3):277–291, 2019.

Guilherme E Vieira, Jeffrey W Herrmann, and Edward Lin. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of scheduling*, 6(1):39–62, 2003.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in neural information processing systems*, volume 35, pp. 24611–24624, 2022.

Zhi Zheng, Shunyu Yao, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Ke Tang. DPN: Decoupling partition and navigation for neural solvers of min-max vehicle routing problems. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=ar174skI9u.

Zefang Zong, Meng Zheng, Yong Li, and Depeng Jin. MAPDP: Cooperative multi-agent reinforcement learning to solve pickup and delivery problems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 9980–9988, 2022.

A Problem Definitions

This appendix defines the optimization problems and residual deterministic models (i.e., models of the remaining problem at each decision step) used by the baselines.

A.1 MACTP

Problem Definition. MACTP generalizes the Canadian Traveler Problem (Papadimitriou & Yannakakis, 1991) to a cooperative team that starts from distinct nodes and covers shared goals on a connected undirected graph with positive traversal costs. Agents try to reach each goal by traversing a graph, but a subset of edges may be blocked, with a given probability, and the true state of the edge is only revealed when an agent reaches one of its endpoints. The blockage probability of each edge is determined at instance generation time. In order to fulfill a goal, an agent must take the action to claim it, which retires that agent. Agents may pass through a goal without claiming it, and may pass through an already claimed goal. The episode succeeds once all goals have been claimed. A representative MACTP example was shown in Figure 1 in the main text.

We formulate the optimization problem as finding a minimum-cost collection of walks that jointly cover the remaining goals.

Indices and Sets

- $\mathcal{N} = \{1, \dots, n\}$ denotes the nodes, and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ contains the directed arcs induced by the usable undirected edges, with cost $w_{vv'} > 0$.
- $\mathcal{D} \subseteq \mathcal{N}$: the set of goal nodes.
- $i \in \mathcal{M}$: agents represented in the current planning problem, with agent i starting its residual walk at s_i.
- $\mathcal{D}^{\text{cov}} \subseteq \mathcal{D}$: goals still requiring coverage, with $G_t = |\mathcal{D}^{\text{cov}}|$.

Decision Variables

- $x_{i,(v,v')} \in \mathbb{Z}_{\geq 0}$: number of traversals of arc (v, v') by agent i.
- $f_{i,(v,v')} \geq 0$: auxiliary connectivity flow carried only on arcs used by agent i.
- $y_{i,v} \in \{0, 1\}$ indicates whether agent i is credited with covering v and is fixed to zero for $v \notin \mathcal{D}^{\text{cov}}$.
- $e_{i,v} \in \{0, 1\}$: whether agent i ends its residual walk at $v \in \mathcal{N}$.

Objective Function

$$\min \quad \sum_{i \in \mathcal{M}} \sum_{(v,v') \in \mathcal{E}} w_{vv'} x_{i,(v,v')} \tag{11}$$

Subject to

$$\sum_{v \in \mathcal{N}} e_{i,v} = 1, \qquad \forall i \in \mathcal{M} \tag{12}$$

$$\sum_{(v,v') \in \mathcal{E}} x_{i,(v,v')} - \sum_{(v',v) \in \mathcal{E}} x_{i,(v',v)} = \mathbf{1}[v = s_i] - e_{i,v}, \qquad \forall i \in \mathcal{M},\ v \in \mathcal{N} \tag{13}$$

$$\sum_{(v,v') \in \mathcal{E}} f_{i,(v,v')} - \sum_{(v',v) \in \mathcal{E}} f_{i,(v',v)} = \Big(\sum_{g \in \mathcal{D}^{\text{cov}}} y_{i,g} \Big) \mathbf{1}[v = s_i] - y_{i,v} \mathbf{1}[v \in \mathcal{D}^{\text{cov}}], \quad \forall i \in \mathcal{M},\ v \in \mathcal{N} \tag{14}$$

$$f_{i,(v,v')} \leq G_t x_{i,(v,v')}, \qquad \forall i \in \mathcal{M},\ (v, v') \in \mathcal{E} \tag{15}$$

$$\sum_{i \in \mathcal{M}} y_{i,v} = 1, \qquad \forall v \in \mathcal{D}^{\text{cov}} \tag{16}$$

Constraints and Online Replanning. The degree constraints define one walk per agent, while the auxiliary flow certifies that every credited goal lies on that walk. During online replanning, revealed

blocked edges are removed, whereas unrevealed edges remain available under their optimistic status. If the revealed graph makes complete coverage infeasible, we replace Eq. (16) with $\sum_i y_{i,v} + u_v = 1$ and lexicographically minimize the number of uncovered goals before travel cost.

MPOMDP Formulation. In the notation of Section 3, the latent variable ξ defines the blockage probability of the edges, while the graph state $\mathcal{G}_t$ is the known graph state. Node features contain goal coverage and claim status. Edge features contain traversal cost, blocking probability, and the currently known status, while each agent carries its present location as an agent feature.

The active set $\mathcal{M}_t$ contains the whole team, although retired agents can only remain at their current nodes and live agents move only through deterministic or revealed-open edges. An agent may retire by claiming an unclaimed goal, where unit capacity and unit claim demand enforce exclusivity, while non-goal nodes impose no shared constraint.

The step cost $C(\mathcal{G}_t, \mathbf{a}_t, \xi)$ is the team's total traversal cost, with retired agents incurring zero cost. Episodes end after all goals are covered or all agents retire, and truncated episodes additionally receive an estimate of the remaining completion cost.

A.2 DFJSP

Problem Definition. In DFJSP, every job follows an ordered chain of operations, and each operation is eligible for a specified subset of machines. Some machines are unavailable from the beginning of an episode, but their condition remains hidden until the first attempted dispatch. A failed dispatch leaves the operation unscheduled and delays its job by a fixed amount, after which the broken machine is removed from future decisions. Failures are correlated through latent groups whose memberships and priors are observed, while the group condition and each machine-specific outcome remain hidden. Machines act as agents and may schedule operations before becoming idle, with processing beginning once both the selected machine and job are available. The objective function is the makespan of the resulting schedule. Figure 5 visualizes an illustrative DFJSP instance.

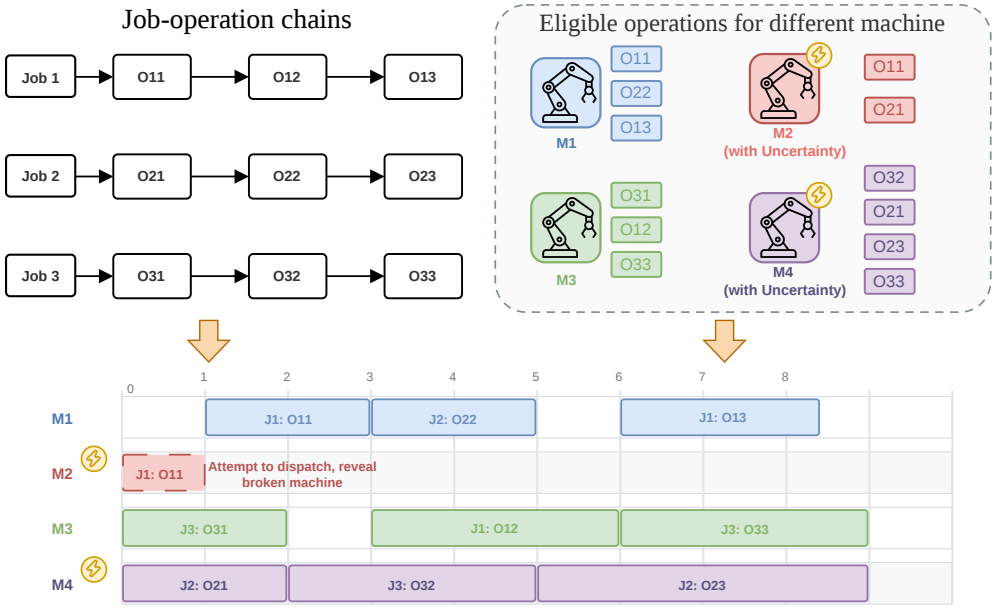


Figure 5: Illustration of the DFJSP task. The upper panel pairs ordered job chains with machine-specific eligibility sets, where the lightning bolts mark machines that may have latent breakdowns. The lower panel gives a machine-wise schedule, where the dashed block records a failed dispatch to a broken machine and the solid block shows successful operation dispatch.

We formulate the optimization problem as a standard disjunctive FJSP over the remaining operations and the machines currently considered usable (Özgüven et al., 2010).

Indices and Sets

- $j \in \mathcal{J}$ indexes jobs, $\mathrm{chain}(j)$ is the ordered operation chain of job j, and $\mathcal{O}_t$ contains the operations not yet scheduled at replanning step t.
- $\mathrm{Elig}_t(o) \subseteq \mathcal{I}$: machines currently considered usable for operation o, with processing time $p_{o,i} > 0$ for $i \in \mathrm{Elig}_t(o)$.
- $r_{o,t}$ is the earliest time permitted by the current job state, and $q_{i,t}$ is the next time at which machine i becomes available.

Decision Variables

- $x_{o,i} \in \{0, 1\}$: whether operation o is assigned to machine $i \in \mathrm{Elig}_t(o)$.
- $s_o \geq 0$ is the start time of operation o, and $d_o = \sum_{i \in \mathrm{Elig}_t(o)} p_{o,i} x_{o,i}$ is its assigned duration.
- $z_{o,o'} \in \{0, 1\}$: relative order of two distinct operations that may share a machine, defined once for each unordered pair $o < o'$.
- $C_{\max} \geq 0$ is the makespan, and B is any valid upper bound on the completion time of the residual schedule.

Objective Function

$$\min \quad C_{\max} \tag{17}$$

Subject to

$$\sum_{i \in \mathrm{Elig}_t(o)} x_{o,i} = 1, \qquad \forall o \in \mathcal{O}_t \tag{18}$$

$$C_{\max} \geq s_o + d_o, \qquad \forall o \in \mathcal{O}_t \tag{19}$$

$$s_{o'} \geq s_o + d_o, \qquad \forall (o, o') \text{ consecutive in a residual job chain} \tag{20}$$

$$s_o \geq r_{o,t}, \qquad \forall o \in \mathcal{O}_t \tag{21}$$

$$s_o \geq q_{i,t} - B(1 - x_{o,i}), \qquad \forall o \in \mathcal{O}_t,\ i \in \mathrm{Elig}_t(o) \tag{22}$$

$$s_{o'} \geq s_o + d_o - B(1 - z_{o,o'}) - B(2 - x_{o,i} - x_{o',i}), \qquad \forall o < o',\ i \in \mathrm{Elig}_t(o) \cap \mathrm{Elig}_t(o') \tag{23}$$

$$s_o \geq s_{o'} + d_{o'} - B z_{o,o'} - B(2 - x_{o,i} - x_{o',i}), \qquad \forall o < o',\ i \in \mathrm{Elig}_t(o) \cap \mathrm{Elig}_t(o') \tag{24}$$

Constraints and Online Replanning. Each remaining operation is assigned once and must follow its job predecessor, while the release constraints preserve the current job and machine timelines. The pairwise disjunctions prevent operations on the same machine from overlapping. At each replanning step, completed operations and revealed-broken machines are removed, whereas unrevealed machines remain provisionally available with their known processing times.

MPOMDP Formulation. The latent variable ξ records the condition of every uncertain machine, with correlations induced by group-level variables. Each element of $\mathcal{G}_t$ represents a job and records its next unscheduled operation and earliest feasible start time. Machine features contain availability, group membership, and the current condition estimate, while an eligibility edge carries the processing time for the job's next operation.

The active set $\mathcal{M}_t$ contains every machine not known to be broken, including busy machines, which may either dispatch an eligible job or wait. A scheduled operation starts no earlier than the release times of both participants, while unit job capacity permits at most one dispatch to each job per round.

The step cost $C(\mathcal{G}_t, \mathbf{a}_t, \xi)$ is the increase in makespan, so its undiscounted sum equals $C_{\max}$ for a completed schedule. A truncated episode additionally receives a penalty of processing all remaining operations serially on their slowest available machines, providing a pessimistic completion cost that prevents stalling from being beneficial.

A.3 NCFFSP

Problem Definition. In NCFFSP, jobs pass through a common sequence of stages, and each stage is served by parallel machines whose processing times depend on both the job and the selected machine (Kwon et al., 2021). Some machines also have hidden condition factors that scale their processing times and are correlated through latent batches. Batch membership and its prior are observed, but a machine's realized factor appears only after its first dispatch. Unlike DFJSP, every dispatch succeeds because uncertainty changes processing duration rather than machine eligibility, and busy machines may schedule jobs that begin once both participants are available. The objective is the final makespan, and non-clairvoyance here refers to the initially unknown machine-dependent duration factors (Motwani et al., 1994). Figure 6 illustrates an NCFFSP workflow.

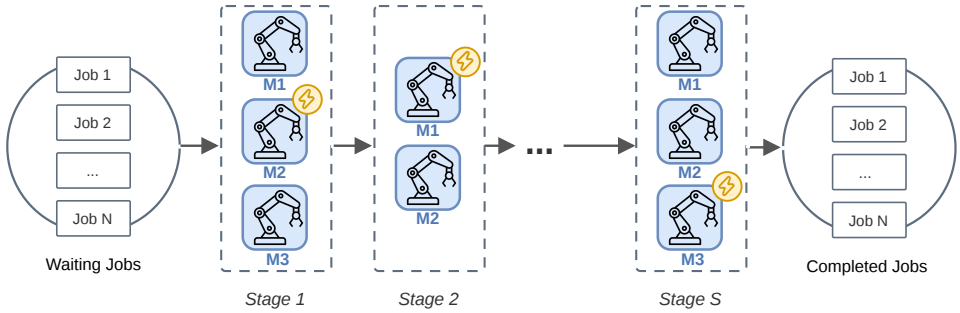


Figure 6: Illustration of the NCFFSP task. Jobs operate from left to right through consecutive parallel machine stages, where the lightning bolts mark machines that may have unrevealed duration factors.

We formulate the optimization problem over the remaining job stages as a flexible flow-shop model using the current processing times. Assignment, stage-precedence, release-time, and machine non-overlap constraints are used.

Indices and Sets

- $\ell \in \{1, \ldots, S\}$ indexes stages, and $\mathcal{I}_\ell$ contains the machines serving stage ℓ.
- $j, j' \in \mathcal{J}_t$: jobs with at least one stage still to be scheduled in the residual problem.
- $\bar{p}_{\ell ji} > 0$: nominal processing time of job j on machine $i \in \mathcal{I}_\ell$.
- $\xi_{\ell i}$: hidden condition factor of machine i. Before revelation, $p_{\ell ji,t} = \bar{p}_{\ell ji}$, while afterwards it is the positive rounded value of $\bar{p}_{\ell ji}\xi_{\ell i}$.
- $r_{\ell j,t}$ is the current release-time floor for job j at stage ℓ, and $q_{i,t}$ is the next time at which machine i becomes available.

Decision Variables

- $x_{\ell ji} \in \{0, 1\}$: whether job j is assigned to machine $i \in \mathcal{I}_\ell$ at stage ℓ.
- $s_{\ell j} \geq 0$ is the start time, and $d_{\ell j} = \sum_{i \in \mathcal{I}_\ell} p_{\ell ji,t} x_{\ell ji}$ is the assigned duration.
- $z_{\ell jj'} \in \{0, 1\}$: relative order of distinct jobs $j < j'$ that may share a machine at stage ℓ.
- $C_{\max} \geq 0$ is the makespan, and B is any valid upper bound on the completion time of the residual schedule.

Objective Function

$$\min \quad C_{\max} \tag{25}$$

Subject to

$$\sum_{i \in \mathcal{I}_\ell} x_{\ell j i} = 1, \qquad \forall j,\ \ell \text{ remaining for } j \tag{26}$$

$$C_{\max} \geq s_{\ell j} + d_{\ell j}, \qquad \forall j,\ \ell \text{ remaining for } j \tag{27}$$

$$s_{\ell+1,j} \geq s_{\ell j} + d_{\ell j}, \qquad \forall j,\ \ell < S \text{ remaining for } j \tag{28}$$

$$s_{\ell j} \geq r_{\ell j,t}, \qquad \forall j,\ \ell \text{ remaining for } j \tag{29}$$

$$s_{\ell j} \geq q_{i,t} - B(1 - x_{\ell j i}), \qquad \forall j,\ \ell \text{ remaining for } j,\ i \in \mathcal{I}_\ell \tag{30}$$

$$s_{\ell j'} \geq s_{\ell j} + d_{\ell j} - B(1 - z_{\ell j j'}) - B(2 - x_{\ell j i} - x_{\ell j' i}), \qquad \forall \ell \text{ remaining for } j, j',\ j < j',\ i \in \mathcal{I}_\ell \tag{31}$$

$$s_{\ell j} \geq s_{\ell j'} + d_{\ell j'} - B z_{\ell j j'} - B(2 - x_{\ell j i} - x_{\ell j' i}), \qquad \forall \ell \text{ remaining for } j, j',\ j < j',\ i \in \mathcal{I}_\ell \tag{32}$$

Constraints and Online Replanning The assignment and non-overlap constraints play the same roles as in DFJSP, while Eq. (28) enforces the shared stage order. The release bounds preserve the current job and machine timelines, while online replanning uses nominal durations until a machine is revealed and realized durations thereafter. Every machine remains eligible for future assignments.

MPOMDP Formulation. NCFFSP uses the job-machine graph interface of DFJSP, but its latent variable ξ contains machine-dependent duration factors rather than binary failure states, and each job element records its next stage and earliest start time. Machine features store the served stage, current availability, batch membership, and either the prior or the revealed factor, while each eligibility edge carries the effective processing time known at that step.

Every machine remains in $\mathcal{M}_t$, including those that are currently busy, and may either dispatch a job at its stage or wait. Each job has unit capacity within a round, and a dispatch directed to that job contributes unit demand, while the scheduled start time respects both participants' release times and prevents assignments on the same machine from overlapping.

The step cost $C(\mathcal{G}_t, \mathbf{a}_t, \xi)$ is the increase in makespan, whose undiscounted sum equals the final makespan for a completed schedule. A truncated episode additionally receives a penalty of the makespan required to complete the schedule from the truncated state under a uniformly random legal policy. This penalty is separate from the NCFFSP objective and ensures that giving up is no better than acting randomly.

B Experimental Details

This appendix lists the baselines evaluated in Section 5 and the licenses of their corresponding code implementations. We also describe how instances are generated for each of the three problems, the training hyperparameters for every learning-based method, and the hardware and software used to run all experiments.

B.1 Baselines

Classic Solvers. The Gurobi and OR-Tools classic solvers are ran in two modes. The baselines marked *Oracle* in the results solve complete realizations using privileged access to the uncertainty with a time limit of 10 minutes. The *Replan* baselines repeatedly solve the remaining problem under the information available at each decision step, with a time limit of 2 seconds for each re-solve.

Gurobi. Gurobi (Gurobi Optimization, LLC, 2026) solves the mixed-integer linear programs in Appendix A using branch-and-cut. We retain the best feasible solution found for each instance under the time budget and report the mean objective value across the evaluation set.

OR-Tools. OR-Tools (Perron & Furnon) provides a complementary CP-SAT implementation under the same information settings and evaluation configurations.

Graph-based MARL Baselines. MAPPO, IPPO, VDN, and DCG use matched graph encoders over the common observation interface, and their per-agent networks share parameters within the team while the flow of information among agents follows the centralized or decentralized structure of each algorithm.

Table 2: Summary of licenses for baselines.

Resource	Type	License
Gurobi (Gurobi Optimization, LLC, 2026)	Solver (commercial)	Commercial, academic license
OR-Tools (Perron & Furnon)	Solver (open-source library)	Apache License 2.0
BenchMARL (Bettini et al., 2024)	Library (MAPPO, IPPO, and VDN)	MIT License
DCG (Böhmer et al., 2020)	Code	Apache License 2.0
PARCO (Berto et al., 2025)	Code	MIT License
MACSIM (Luttmann & Xie, 2026)	Code	Apache License 2.0

MAPPO. Multi-Agent PPO (Yu et al., 2022) uses a shared-parameter decentralized actor and a centralized critic, so each actor conditions on the common graph information and its own agent features while the critic aggregates information across the team. Both networks are trained with the clipped PPO objective and generalized advantage estimation.

IPPO. Independent PPO (De Witt et al., 2020) uses the same shared-parameter decentralized actor as MAPPO, but replaces the centralized critic with a decentralized value function. The critic produces one estimate per agent from that agent's observation, and training uses the same clipped PPO objective and advantage estimator.

VDN. Value Decomposition Networks (Sunehag et al., 2018) learn shared per-agent action-value functions and select each agent's action independently, after which the selected utilities are summed into a team value, $Q_{\text{tot}} = \sum_i Q_i$, trained from a common temporal-difference target.

DCG. Deep Coordination Graphs (Böhmer et al., 2020) combine per-agent utility terms with pairwise payoff terms defined on a fixed, fully connected coordination graph, and at each step max-plus message passing approximately maximizes the resulting joint action-value function to select the agents' actions.

PARCO. PARCO (Berto et al., 2025) is a parallel autoregressive neural solver in which a shared transformer encodes the instance and communication layers exchange context among agent queries. A multiple-pointer decoder proposes simultaneous actions before a learned priority rule resolves conflicts and assigns fallback actions to displaced agents.

MACSIM. MACSIM (Luttmann & Xie, 2026) also constructs multi-agent solutions autoregressively from a shared instance representation. To produce a conflict-free set of simultaneous actions, it samples agent-task pairs without replacement from a joint categorical distribution at each construction step.

Neural Solver Adaptation. We implement task wrappers for PARCO and MACSIM through their respective environment interfaces while preserving the instances, transition rules, action constraints, and objectives used to evaluate the other methods.

Solution Decoding Strategies. We evaluate PARCO, MACSIM, and PACT under two solution decoding strategies: *greedy* and *sampling*. Greedy decoding uses one deterministic rollout, while sampling decoding generates 128 independent stochastic rollouts per method and instance and retains that method's best realized solution (Berto et al., 2025; Luttmann & Xie, 2026). For greedy decoding, PARCO and MACSIM select the argmax of their action distributions, while in PACT the internal candidate plan generation, virtual rollout, and plan selection are all made deterministic.

B.2 Code and Licenses

Table 2 lists the external solvers and codebases used to implement the baselines above, together with their licenses.

B.3 Data Generation

MACTP. Following SALSA-CLRS (Minder et al., 2023), each instance uses a sparse, connected Erdős–Rényi graph over n nodes, with agent starts and goals sampled uniformly at random and positive traversal costs assigned to its edges. In the reference configuration, 25% of the edges are stochastic, and their traversability is sampled at generation time but remains hidden until encountered, as described in Appendix A.1.

DFJSP. We follow the synthetic FJSP generation procedure of Song et al. (2022): each job contains four to six ordered operations, each operation is assigned a random subset of eligible machines, and its machine-dependent processing times vary within 20% of an operation-level mean. Each machine is initially designated stochastic with probability 0.6, after which a feasibility repair ensures that every operation retains at least one always-available eligible machine. Stochastic machines have fixed hidden availability and correlated breakdown probabilities within small machine groups, so observing one machine can inform the condition of others. Availability is revealed only when a job is first dispatched to the machine, and the resulting transition follows Appendix A.2.

NCFFSP. Each instance contains N jobs that pass through S processing stages, with several interchangeable machines available at each stage. Following Kwon et al. (2021), nominal job–machine processing times are sampled independently and uniformly from the integers $\{2, \ldots, 9\}$. Each real machine is independently designated stochastic with probability 0.6 and receives a fixed hidden condition factor generated from a group-correlated latent variable $u \in [0, 1]$ through $\xi = 0.8 + 7.2u^4$. The factor is revealed on the machine's first dispatch and scales its processing times thereafter, as described in Appendix A.3.

B.4 Training Configuration

PACT and the four MARL baselines use a common number of training steps, data-collection scale, optimizer, and graph-encoder configuration across all three tasks. Table 3 reports these shared settings together with the update parameters specific to the on-policy and value-based families. We evaluate checkpoints on the same 64 evaluation episodes every 20,000 steps, and each configuration is trained with 10 random seeds.

PACT-specific Settings. PACT uses two coordination rounds in every task and is evaluated with $K = 4$ candidate plans per agent. During training, K is sampled from $\{2, 4, 8\}$, while the planning horizon is $H_p = 4$ on MACTP and $H_p = 1$ on DFJSP and NCFFSP. All three decision groups use a clipping range of 0.2. The proposal and rollout groups use an entropy coefficient of 0.001, while the selection group uses 0.01.

Neural Combinatorial Optimization Baselines. PARCO and MACSIM retain their original training objectives and model configurations, with PARCO using its rollout-based objective and MACSIM using self-improvement. Both methods use Adam with learning rate 1×10^{-4} and are trained for 50 epochs on each task.

Table 3: Training configuration for PACT and the MARL baselines.

Hyperparameter	Value
Shared settings	
Training steps	500,000
Validation interval	20,000 steps
Validation episodes	64
Random seeds	10
Optimizer	Adam
Learning rate	5×10^{-5}
Steps per batch	2,000
Parallel environments	4
Maximum gradient norm	5
Encoder layers	2
Hidden dimension	64
Aggregation	Weighted mean
Discount factor	0.99 (MACTP); 1.0 (DFJSP and NCFFSP)
PPO settings (PACT, MAPPO, and IPPO)	
Update epochs	4
Minibatch size	128
Clipping coefficient	0.2
GAE parameter λ	0.95
Entropy coefficient (MAPPO and IPPO)	0.01
Value-loss coefficient	1.0
Value-based settings (VDN and DCG)	
Updates per batch	200
Minibatch size	256
Replay-buffer capacity	30,000
Target-update coefficient	0.005
Initial exploration rate ϵ	0.8
DCG-specific settings	
Payoff rank	8
Max-plus iterations	4
PACT-specific settings	
Score step bound η	1.0 (fixed)
Selection temperature τ	Learned, initialized to 1.0, $\tau \geq 0.25$
Coordination MPNN layers	1
Coordination hidden dimension	64

B.5 Compute and Software

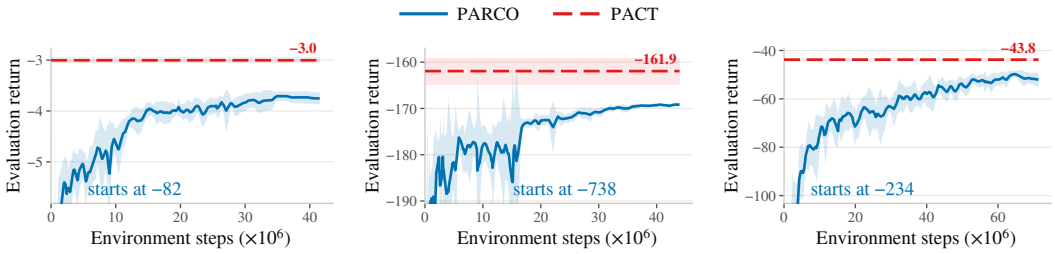
All learning-based methods were trained and evaluated on NVIDIA L40S GPUs, while the Gurobi and OR-Tools baselines were run on an Intel Core Ultra 7 265HX CPU. Experiments used Python 3.11, PyTorch 2.9.1, TorchRL 0.10.1, TensorDict 0.10.0, PyTorch Geometric 2.6.1, NumPy 2.1.3, and NetworkX 3.6.1. The optimization baselines used Gurobi 13.0.3 and OR-Tools 9.12.4544.

C Additional Experimental Results

C.1 Training Curves for PARCO and MACSIM

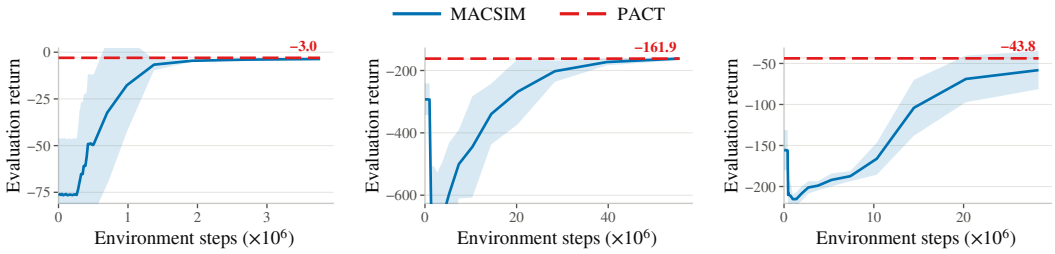
Figures 7 and 8 show the validation returns of PARCO and MACSIM during training on the three DMACO problems. Curves show the mean across ten seeds, with shaded regions indicating one standard deviation. The horizontal red dashed line marks PACT's highest mean validation return during training, and the red shaded band shows one standard deviation across seeds at the corresponding checkpoint. PACT achieves a higher best mean validation return than PARCO across all

three problems and outperforms MACSIM on NCFFSP. While it converges to returns comparable to PACT on MACTP and DFJSP, MACSIM consistently requires substantially more training steps across all tasks. Overall, PACT demonstrates a significantly improved trade-off between sample efficiency and validation performance. Furthermore, despite occasional matching validation performance in-distribution, we nevertheless highlight the significantly higher performance on larger ZST instances reported in the main text.



(a) MACTP (b) DFJSP (c) NCFFSP

Figure 7: Validation return during training for PARCO on the three DMACO problems.



(a) MACTP (b) DFJSP (c) NCFFSP

Figure 8: Validation return during training for MACSIM on the three DMACO problems.

C.2 Complete Results for Learning-Based Methods

We evaluate each method on 100 instances for each of 10 independently trained seeds and report the mean across seeds. Success is the percentage of successfully completed instances, cost is the mean negated episodic return over all episodes, including penalties from failed or truncated episodes, whereas objective is the mean negated episodic return over successfully completed episodes. Performance ratio is cost divided by the mean cost of PACT on the same problem and test configuration within the same evaluation group: greedy methods use greedy PACT as the reference, and sampling methods use sampling PACT. PACT has a ratio of one within each group, and lower values are better. All ratios are computed from the cost values as reported in the table. Time reports the end-to-end runtime per instance in seconds. Each entry reports the mean and one standard deviation across the ten training seeds; performance ratio is computed from the mean cost and reported without a standard deviation. N/A indicates that the method does not support the configuration.

Tables 4, 5, and 6 report these metrics under greedy and best-of-128 sampling evaluation.

Table 4: Complete results for learning-based methods on MACTP. The first configuration is in-distribution (ID), and the remaining configurations are evaluated zero-shot (ZST).

Method	Evaluation	Success (%) ↑	Cost ↓	Objective ↓	Perf. ratio ↓	Time (s) ↓
$16 \times 4 \times 6$ (ID)						
MAPPO	Greedy	96.8±2.1	3.83±0.25	3.53±0.06	1.046	0.027±0.030
IPPO	Greedy	97.1±2.0	3.92±0.36	3.57±0.11	1.071	0.015±0.007
VDN	Greedy	98.9±0.9	**3.52**±0.20	**3.33**±0.04	0.962	0.012±0.004
DCG	Greedy	**99.3**±0.7	3.68±0.47	3.55±0.26	1.005	0.011±0.003
PARCO	Greedy	98.9±1.2	4.56±0.23	4.34±0.13	1.246	0.004±0.002
MACSIM	Greedy	89.9±3.7	7.36±0.68	5.55±0.35	2.011	0.038±0.021
PACT	Greedy	99.1±1.4	3.66±0.18	3.43±0.07	1.000	0.056±0.015
PARCO	Sampling	**100.0**±0.0	3.35±0.04	3.35±0.04	1.159	0.094±0.016
MACSIM	Sampling	**100.0**±0.0	3.27±0.09	3.27±0.09	1.131	0.48±0.07
PACT	Sampling	**100.0**±0.0	**2.89**±0.02	**2.89**±0.02	1.000	5.27±0.49
$32 \times 6 \times 10$ (ZST)						
MAPPO	Greedy	98.1±1.8	6.78±0.68	6.19±0.19	1.171	0.020±0.007
IPPO	Greedy	98.8±1.4	6.84±0.76	6.27±0.22	1.181	0.033±0.052
VDN	Greedy	**99.6**±1.0	5.84±0.40	5.69±0.23	1.009	0.019±0.019
DCG	Greedy	98.1±1.4	8.44±4.49	7.84±4.36	1.458	0.048±0.029
PARCO	Greedy	95.0±2.4	10.8±1.8	7.67±0.25	1.865	0.017±0.005
MACSIM	Greedy	95.2±5.1	12.7±1.3	11.0±0.7	2.193	0.035±0.010
PACT	Greedy	**99.6**±0.8	**5.79**±0.38	**5.68**±0.13	1.000	0.064±0.053
PARCO	Sampling	**100.0**±0.0	5.50±0.06	5.50±0.06	1.180	0.15±0.06
MACSIM	Sampling	**100.0**±0.0	6.32±0.42	6.32±0.42	1.356	0.68±0.28
PACT	Sampling	**100.0**±0.0	**4.66**±0.05	**4.66**±0.05	1.000	5.45±0.67
$128 \times 12 \times 24$ (ZST)						
MAPPO	Greedy	95.7±5.5	32.8±16.4	20.6±3.8	1.763	0.79±0.66
IPPO	Greedy	97.2±2.4	28.6±10.3	21.3±6.1	1.538	0.63±0.48
VDN	Greedy	**99.9**±0.3	20.7±4.5	20.3±4.1	1.113	0.38±0.15
DCG	Greedy	93.3±17.8	70.0±101.0	52.9±69.2	3.763	0.99±1.40
PARCO	Greedy	41.2±24.2	195.1±89.4	48.9±36.0	10.489	0.11±0.04
MACSIM	Greedy	99.6±0.7	34.8±3.4	33.4±3.0	1.871	0.047±0.032
PACT	Greedy	99.7±0.7	**18.6**±3.1	**17.8**±0.9	1.000	0.34±0.18
PARCO	Sampling	99.7±0.7	17.8±5.2	17.6±4.6	1.148	2.67±1.95
MACSIM	Sampling	**100.0**±0.0	24.0±2.9	24.0±2.9	1.548	1.91±1.63
PACT	Sampling	**100.0**±0.0	**15.5**±0.9	**15.5**±0.9	1.000	10.5±1.6
$512 \times 24 \times 48$ (ZST)						
MAPPO	Greedy	OOM	OOM	OOM	OOM	OOM
IPPO	Greedy	OOM	OOM	OOM	OOM	OOM
VDN	Greedy	OOM	OOM	OOM	OOM	OOM
DCG	Greedy	OOM	OOM	OOM	OOM	OOM
PARCO	Greedy	4.2±8.4	1722.0±555.4	225.5±200.4	23.817	0.35±0.06
MACSIM	Greedy	**100.0**±0.0	90.2±3.1	90.2±3.1	1.248	0.18±0.01
PACT	Greedy	99.0±2.5	**72.3**±42.8	**46.2**±4.0	1.000	2.61±2.71
PARCO	Sampling	86.3±10.3	372.6±267.9	112.4±116.5	7.408	9.83±12.04
MACSIM	Sampling	**100.0**±0.0	84.2±13.3	84.2±13.3	1.674	42.0±8.9
PACT	Sampling	**100.0**±0.0	**50.3**±5.8	**50.3**±5.8	1.000	51.9±14.6
$1024 \times 32 \times 64$ (ZST)						
MAPPO	Greedy	OOM	OOM	OOM	OOM	OOM
IPPO	Greedy	OOM	OOM	OOM	OOM	OOM
VDN	Greedy	OOM	OOM	OOM	OOM	OOM
DCG	Greedy	OOM	OOM	OOM	OOM	OOM
PARCO	Greedy	4.5±10.0	3999.2±1871.4	472.0±558.1	46.448	0.73±0.15
MACSIM	Greedy	**100.0**±0.0	131.3±2.2	131.3±2.2	1.525	2.20±0.15
PACT	Greedy	99.6±1.3	**86.1**±29.2	**70.5**±11.1	1.000	11.0±14.7
PARCO	Sampling	71.3±16.2	1341.6±349.6	187.1±214.0	15.971	27.5±27.9
MACSIM	Sampling	**100.0**±0.0	125.8±15.0	125.8±15.0	1.498	381.7±31.0
PACT	Sampling	**100.0**±0.0	**84.0**±10.8	**84.0**±10.8	1.000	136.9±27.9

Table 5: Complete results for learning-based methods on DFJSP. The first configuration is in-distribution (ID), and the remaining configurations are evaluated zero-shot (ZST).

Method	Evaluation	Success (%) ↑	Cost ↓	Objective ↓	Perf. ratio ↓	Time (s) ↓
$20 \times 8 \times 6$ (ID)						
MAPPO	Greedy	**100.0**±0.0	170.2±3.2	170.2±3.2	1.047	0.022±0.008
IPPO	Greedy	96.1±9.4	207.7±83.7	176.2±7.1	1.278	0.051±0.017
VDN	Greedy	54.4±48.2	632.1±486.7	181.7±6.4	3.890	0.056±0.028
DCG	Greedy	15.1±32.1	986.6±411.5	187.8±14.6	6.071	0.088±0.004
PARCO	Greedy	**100.0**±0.0	170.5±0.4	170.5±0.4	1.049	0.004±0.001
MACSIM	Greedy	54.7±20.3	608.3±212.0	166.1±4.3	3.743	0.18±0.03
PACT	Greedy	**100.0**±0.0	**162.5**±2.9	**162.5**±2.9	1.000	0.093±0.042
PARCO	Sampling	**100.0**±0.0	165.0±0.3	165.0±0.3	1.058	0.087±0.001
MACSIM	Sampling	**100.0**±0.0	156.2±1.1	156.2±1.1	1.002	0.20±0.09
PACT	Sampling	**100.0**±0.0	**155.9**±1.4	**155.9**±1.4	1.000	10.5±3.7
$30 \times 14 \times 6$ (ZST)						
MAPPO	Greedy	**100.0**±0.0	148.4±3.2	148.4±3.2	1.049	0.021±0.009
IPPO	Greedy	99.7±0.7	156.1±12.5	152.4±4.8	1.103	0.054±0.017
VDN	Greedy	66.5±46.3	688.7±748.7	158.3±5.6	4.867	0.076±0.049
DCG	Greedy	13.4±28.1	1476.8±604.8	217.6±76.6	10.437	0.13±<0.01
PARCO	Greedy	**100.0**±0.0	146.0±0.5	146.0±0.5	1.032	0.004±0.001
MACSIM	Greedy	70.1±21.7	596.9±311.7	**141.5**±4.3	4.218	0.21±0.05
PACT	Greedy	**100.0**±0.0	**141.5**±3.6	**141.5**±3.6	1.000	0.10±0.05
PARCO	Sampling	**100.0**±0.0	142.8±0.5	142.8±0.5	1.065	0.096±0.001
MACSIM	Sampling	**100.0**±0.0	136.4±1.2	136.4±1.2	1.017	0.25±0.09
PACT	Sampling	**100.0**±0.0	**134.1**±1.9	**134.1**±1.9	1.000	10.5±4.3
$50 \times 16 \times 8$ (ZST)						
MAPPO	Greedy	**100.0**±0.0	245.1±5.9	245.1±5.9	1.051	0.070±0.033
IPPO	Greedy	98.7±3.0	295.6±86.2	261.6±11.0	1.268	0.24±0.11
VDN	Greedy	57.7±50.0	1641.3±1647.5	256.2±18.6	7.038	0.33±0.27
DCG	Greedy	7.4±15.9	3089.9±1041.2	335.3±71.8	13.250	0.61±0.01
PARCO	Greedy	99.7±0.9	258.4±2.7	253.7±2.0	1.108	0.015±0.024
MACSIM	Greedy	60.2±29.8	1241.3±195.5	247.3±9.2	5.323	0.48±0.13
PACT	Greedy	**100.0**±0.0	**233.2**±3.6	**233.2**±3.6	1.000	0.25±0.15
PARCO	Sampling	**100.0**±0.0	249.7±1.7	249.7±1.7	1.074	0.18±0.01
MACSIM	Sampling	**100.0**±0.0	242.3±1.8	242.3±1.8	1.043	0.93±0.93
PACT	Sampling	**100.0**±0.0	**232.4**±2.6	**232.4**±2.6	1.000	23.6±11.0
$80 \times 20 \times 8$ (ZST)						
MAPPO	Greedy	**100.0**±0.0	299.0±9.8	299.0±9.8	1.053	0.24±0.12
IPPO	Greedy	98.6±3.5	380.1±162.5	319.9±16.3	1.339	0.87±0.49
VDN	Greedy	59.6±51.3	2498.4±2780.2	318.3±18.6	8.800	1.27±1.12
DCG	Greedy	5.4±11.4	5017.7±1604.1	452.9±148.6	17.674	2.72±0.02
PARCO	Greedy	97.4±5.9	336.2±32.6	314.6±4.6	1.184	0.054±0.084
MACSIM	Greedy	58.0±32.2	1305.2±1130.6	312.5±11.6	4.597	0.84±0.24
PACT	Greedy	**100.0**±0.0	**283.9**±4.7	**283.9**±4.7	1.000	0.77±0.50
PARCO	Sampling	**100.0**±0.0	314.5±3.9	314.5±3.9	1.125	0.28±0.03
MACSIM	Sampling	**100.0**±0.0	307.2±2.3	307.2±2.3	1.099	1.83±1.75
PACT	Sampling	**100.0**±0.0	**279.6**±3.5	**279.6**±3.5	1.000	54.4±28.7
$120 \times 24 \times 10$ (ZST)						
MAPPO	Greedy	**100.0**±0.0	429.2±20.5	429.2±20.5	1.077	0.84±0.45
IPPO	Greedy	98.5±4.4	584.4±354.4	469.7±30.9	1.467	3.45±2.43
VDN	Greedy	59.9±51.6	4311.0±4961.6	467.5±27.1	10.821	5.54±5.48
DCG	Greedy	0.5±1.6	9296.8±2400.1	1030.2±1062.3	23.335	13.1±0.1
PARCO	Greedy	7.6±13.7	2578.1±608.2	439.1±23.1	6.471	0.35±0.08
MACSIM	Greedy	61.2±32.3	1866.3±641.7	462.0±19.7	4.684	1.64±0.52
PACT	Greedy	**100.0**±0.0	**398.4**±6.8	**398.4**±6.8	1.000	2.41±1.56
PARCO	Sampling	**100.0**±0.0	490.3±13.6	490.3±13.6	1.237	2.30±2.86
MACSIM	Sampling	**100.0**±0.0	458.3±2.9	458.3±2.9	1.156	5.24±9.66
PACT	Sampling	**100.0**±0.0	**396.5**±4.9	**396.5**±4.9	1.000	112.1±27.3

Table 6: Complete results for learning-based methods on NCFFSP. The first configuration is in-distribution (ID), and the remaining configurations are evaluated zero-shot (ZST).

Method	Evaluation	Success (%) ↑	Cost ↓	Objective ↓	Perf. ratio ↓	Time (s) ↓
$20 \times 4 \times 3$ (ID)						
MAPPO	Greedy	97.4±3.5	46.5±2.3	44.2±1.0	1.120	0.019±0.006
IPPO	Greedy	98.5±1.9	47.3±1.7	45.4±1.5	1.140	0.017±0.004
VDN	Greedy	97.8±2.2	50.0±2.1	48.6±1.5	1.205	0.017±0.005
DCG	Greedy	71.7±31.6	84.1±50.6	56.1±17.0	2.027	0.036±0.011
PARCO	Greedy	99.3±0.7	42.9±0.5	42.5±0.4	1.034	0.005±0.002
MACSIM	Greedy	82.9±32.5	64.7±43.0	45.9±5.9	1.559	0.067±0.022
PACT	Greedy	**100.0**±0.0	**41.5**±0.7	**41.5**±0.7	1.000	0.049±0.019
PARCO	Sampling	**100.0**±0.0	37.3±0.2	37.3±0.2	1.030	0.085±0.002
MACSIM	Sampling	**100.0**±0.0	39.0±1.5	39.0±1.5	1.077	0.21±0.03
PACT	Sampling	**100.0**±0.0	**36.2**±0.6	**36.2**±0.6	1.000	5.35±0.26
$30 \times 6 \times 3$ (ZST)						
MAPPO	Greedy	99.2±1.3	38.0±1.9	37.3±0.8	1.058	0.020±0.007
IPPO	Greedy	99.3±1.3	39.3±1.0	38.5±1.2	1.095	0.020±0.007
VDN	Greedy	96.8±2.8	44.2±3.1	41.3±1.4	1.231	0.027±0.010
DCG	Greedy	58.9±25.5	102.4±53.7	52.3±13.4	2.852	0.069±0.005
PARCO	Greedy	98.8±0.9	49.8±1.2	48.7±1.4	1.387	0.009±0.005
MACSIM	Greedy	91.7±23.3	55.4±46.4	41.2±5.7	1.543	0.057±0.032
PACT	Greedy	**100.0**±0.0	**35.9**±1.9	**35.9**±1.9	1.000	0.063±0.041
PARCO	Sampling	**100.0**±0.0	39.6±0.4	39.6±0.4	1.119	0.099±0.002
MACSIM	Sampling	**100.0**±0.0	36.1±2.3	36.1±2.3	1.020	0.32±0.04
PACT	Sampling	**100.0**±0.0	**35.4**±0.9	**35.4**±0.9	1.000	6.17±0.29
$50 \times 7 \times 3$ (ZST)						
MAPPO	Greedy	99.3±1.3	49.4±2.8	**48.0**±0.7	1.012	0.065±0.027
IPPO	Greedy	98.9±1.9	52.6±2.5	49.7±2.6	1.078	0.070±0.032
VDN	Greedy	93.2±8.3	63.9±12.5	53.5±2.4	1.309	0.12±0.07
DCG	Greedy	59.0±30.0	152.4±79.0	75.4±30.9	3.123	0.26±0.03
PARCO	Greedy	98.8±2.2	66.8±3.5	65.3±1.3	1.369	0.010±0.006
MACSIM	Greedy	86.6±30.5	98.4±104.2	61.3±19.1	2.016	0.11±0.07
PACT	Greedy	**99.9**±0.3	**48.8**±2.4	48.4±1.1	1.000	0.17±0.14
PARCO	Sampling	**100.0**±0.0	57.4±2.3	57.4±2.3	1.281	0.13±0.09
MACSIM	Sampling	**100.0**±0.0	50.7±4.0	50.7±4.0	1.132	0.71±0.09
PACT	Sampling	**100.0**±0.0	**44.8**±1.5	**44.8**±1.5	1.000	11.7±0.6
$80 \times 6 \times 4$ (ZST)						
MAPPO	Greedy	98.2±3.3	94.5±11.8	88.5±1.3	1.133	0.33±0.24
IPPO	Greedy	97.8±4.5	99.8±14.1	91.6±5.0	1.197	0.30±0.24
VDN	Greedy	84.4±11.7	151.0±42.7	98.8±2.9	1.811	1.04±0.42
DCG	Greedy	50.3±18.1	376.7±137.6	141.8±53.5	4.517	1.64±0.03
PARCO	Greedy	N/A	N/A	N/A	N/A	N/A
MACSIM	Greedy	82.7±28.5	212.6±191.5	114.4±27.9	2.549	0.29±0.16
PACT	Greedy	**99.7**±0.7	**83.4**±8.9	**81.1**±2.3	1.000	0.51±0.57
PARCO	Sampling	N/A	N/A	N/A	N/A	N/A
MACSIM	Sampling	**100.0**±0.0	97.3±7.1	97.3±7.1	1.179	2.09±0.66
PACT	Sampling	**100.0**±0.0	**82.5**±2.9	**82.5**±2.9	1.000	32.5±1.6
$120 \times 8 \times 4$ (ZST)						
MAPPO	Greedy	98.4±3.5	93.8±16.1	86.2±2.5	1.082	1.13±1.22
IPPO	Greedy	98.6±3.5	97.1±16.2	90.2±4.4	1.120	1.05±1.15
VDN	Greedy	81.6±13.2	186.1±65.9	96.8±2.8	2.146	4.62±1.60
DCG	Greedy	48.4±27.9	513.9±241.2	186.0±99.1	5.927	6.98±0.35
PARCO	Greedy	N/A	N/A	N/A	N/A	N/A
MACSIM	Greedy	89.1±23.1	213.9±222.1	123.6±29.2	2.467	0.35±0.29
PACT	Greedy	**99.4**±1.1	**86.7**±22.6	**80.4**±2.4	1.000	1.84±2.40
PARCO	Sampling	N/A	N/A	N/A	N/A	N/A
MACSIM	Sampling	**100.0**±0.0	109.6±8.9	109.6±8.9	1.280	3.87±0.62
PACT	Sampling	**100.0**±0.0	**85.6**±4.4	**85.6**±4.4	1.000	97.0±10.0

C.3 Runtime Analysis

Table 7 decomposes PACT's end-to-end runtime into environment stepping, policy inference, and other costs. Policy inference is further divided into encoding, candidate plan generation, trace-graph coordination, plan selection, and remaining policy operations. We report the ID and largest ZST configuration for each task to show how the distribution changes with problem scale.

The runtime profiles reflect both problem scale and the planning horizon H_p. On MACTP, where $H_p = 4$, plan generation accounts for 38.02% on the ID configuration and remains at 42.15% on the largest ZST configuration because each of the K candidates requires a multi-step autoregressive virtual rollout. By contrast, DFJSP and NCFFSP use $H_p = 1$, so plan generation ends after the initial candidate actions and its share falls from 23.82% and 27.49% to 3.74% and 5.27% as the instances grow. Their larger graph states instead make encoding dominant, accounting for 86.37% and 87.76% of runtime. Encoding also increases from 11.91% to 43.77% on MACTP, but its longer rollouts keep plan generation comparably expensive. The relative runtime of coordination decreases from 9.95-13.25% on the ID configurations to 2.81-4.42% on the largest configurations, while selection remains below 1.3% across the evaluated configurations.

Table 7: Breakdown of PACT's end-to-end runtime. Component columns report percentages of the end-to-end runtime, and the final column reports time per instance.

Task	Configuration	Env. step	Policy					Other	Time (s)
			Encoder	Plan	Coord.	Selection	Other		
MACTP	$16 \times 4 \times 6$ (ID)	15.77%	11.91%	38.02%	9.95%	0.5%	15.45%	8.39%	0.056
	$1024 \times 32 \times 64$ (ZST)	5.32%	43.77%	42.15%	2.81%	0.14%	4.5%	1.31%	11.0
DFJSP	$20 \times 8 \times 6$ (ID)	19.04%	15.35%	23.82%	12.92%	1.28%	23.5%	4.08%	0.093
	$120 \times 24 \times 10$ (ZST)	2.38%	86.37%	3.74%	4.42%	0.07%	2.51%	0.53%	2.41
NCFFSP	$20 \times 4 \times 3$ (ID)	19.71%	15.84%	27.49%	13.25%	0.69%	18.59%	4.43%	0.049
	$120 \times 8 \times 4$ (ZST)	2.17%	87.76%	5.27%	3.01%	0.02%	1.26%	0.52%	1.84

C.4 Ablation Study Details

In this section, we provide further details regarding the evaluation protocol and results of the ablation study reported in Section 5.3 in the main text.

All ablation experiments are conducted with greedy evaluation on the largest zero-shot transfer configuration of each problem. We use the PACT model with $K = 4$ candidate plans and $R = 2$ coordination rounds as the reference model. We define the performance ratio as cost divided by the reference model's mean cost on the same problem, and compute the time ratio in the same way using end-to-end runtime. Both ratios equal one for the reference model, and lower values are better. We present the ablation results in Figure 4. For each method or configuration, bars show mean performance ratios across seeds, with error bars indicating one standard deviation, while lines show mean time ratios.

Effect of Coordination. We compare PACT, PACT without coordination, and MAPPO in Figure 4(a). PACT without coordination omits message passing on the plan-augmented trace-graph, so each agent selects its candidate plan independently. PACT achieves the lowest mean performance ratio on all three problems. Removing coordination substantially degrades performance, particularly on MACTP and NCFFSP, and makes it worse than MAPPO on DFJSP and NCFFSP. This indicates that the gains arise from using shared elements to mediate candidate selection rather than from candidate generation alone.

Number of Candidate Plans. We vary the number of candidate plans $K \in \{1, 4, 8, 16\}$ in Figure 4(b). With $K = 1$, each agent exposes only one plan to the trace-graph, so coordination cannot redirect it to an alternative. This setting degrades performance on every problem and produces high variability on NCFFSP, whereas four candidates provide most of the observed improvement. Larger sets yield no consistent gain beyond the variation across seeds. The time ratio reflects both per-step computation and episode length, so longer episodes can offset the savings from fewer candidates. The sharp escalation in the time ratio on MACTP at large K is consistent with $H_p = 4$, compared with $H_p = 1$ on DFJSP and NCFFSP.

Number of Coordination Rounds. We vary the number of coordination rounds $R \in \{1, 2, 3, 4\}$ in Figure 4(c). The first round updates candidate plans from the loads induced by the initial preferences. Subsequent rounds recompute these loads after the preferences change, allowing plans to respond to how other agents have adjusted their selections. One round performs comparably on the scheduling problems but yields a higher mean performance ratio on MACTP, while the second closes this gap. This difference is consistent with MACTP's longer plan traces capturing multi-step interactions, whereas the scheduling tasks use $H_p = 1$. Further rounds provide no consistent improvement, and runtime remains nearly unchanged.

D Detailed Algorithms

Algorithm 1 summarizes how PACT maps the graph observation at one decision step to a joint action. The three algorithms below develop this procedure in sequence: candidate plan generation (Algorithm 2), coordination on the plan-augmented trace-graph (Algorithm 3), and plan selection followed by execution (Algorithm 4). Together, they complement the corresponding descriptions in Section 4.1-Section 4.3 with a more explicit account of the forward pass.

Algorithm 2 Candidate plan generation for agent i (detail of Section 4.1)

Input: graph observation $\mathcal{G}_t$; encodings h_G and σ_i from Eq. (2); agent i; candidate plan count K; plan horizon H_p
Output: candidate plans $\hat{p}_i^1, \ldots, \hat{p}_i^K$ and their traces $\mathrm{Tr}(\hat{p}_i^k)$
$\hat{x}_{i,0}^k \leftarrow$ current state of agent i from $\mathcal{G}_t$ for every k
draw $\hat{a}_{i,t}^1, \ldots, \hat{a}_{i,t}^K$ jointly, without replacement, from $\pi_\theta^{\mathrm{plan}}(\cdot \mid h_G, \sigma_i, \hat{x}_{i,0}^k)$
for $k = 1, \ldots, K$ **do**
  $\hat{x}_{i,1}^k \leftarrow$ rollout state of agent i after $\hat{a}_{i,t}^k$
  for $j = 1, \ldots, H_p - 1$ **do**
    sample $\hat{a}_{i,t+j}^k \sim \pi_\theta^{\mathrm{plan}}(\cdot \mid h_G, \sigma_i, \hat{x}_{i,j}^k)$
    $\hat{x}_{i,j+1}^k \leftarrow$ rollout state after $\hat{a}_{i,t+j}^k$
  end for
  $\hat{p}_i^k \leftarrow (\hat{a}_{i,t}^k, \ldots, \hat{a}_{i,t+H_p-1}^k)$
  $\mathrm{Tr}(\hat{p}_i^k) \leftarrow \{(v, d(v, \hat{a}_{i,t+j}^k), j) : d(v, \hat{a}_{i,t+j}^k) > 0\}$
end for
return $\{\hat{p}_i^k, \mathrm{Tr}(\hat{p}_i^k)\}_{k=1}^K$

Algorithm 3 Coordination on the plan-augmented trace-graph (detail of Section 4.2)

Input: candidate plans $\{\hat{p}_i^k, \mathrm{Tr}(\hat{p}_i^k)\}$ for all $i \in \mathcal{M}_t$, $k = 1, \ldots, K$; coordination rounds R
Output: candidate plan scores $s_i^{(R)}$ for all $i \in \mathcal{M}_t$
$\chi_t \leftarrow \mathcal{G}_t$ with one candidate plan node per $\hat{p}_i^k$ and a trace arc $(\hat{p}_i^k, v)$ for every $(v, d, j) \in \mathrm{Tr}(\hat{p}_i^k)$
$h^{(0)} \leftarrow h_G$ for element nodes and an encoding of the node's trace for candidate plan nodes
$s_i^{(0)}(k) \leftarrow \eta \tanh\big(f_{\mathrm{score}}(h^{(0)}(\hat{p}_i^k))\big), \quad q_i^{(0)} \leftarrow \mathrm{softmax}(s_i^{(0)})$ (Eq. (5))
for $r = 0, \ldots, R-1$ **do**
  for each element v on which some trace places demand **do**
    $L^{(r)}(v) \leftarrow \sum_i \sum_k q_i^{(r)}(k)\, d(v, \hat{p}_i^k)$ (Eq. (6))
    $\tilde{h}^{(r)}(v) = h^{(r)}(v) + W_{\mathrm{load}} \left[\frac{L^{(r)}(v)}{\kappa_t(v)},\ \frac{\mathrm{ReLU}\big(L^{(r)}(v) - (\kappa_t(v) - d_t(v))\big)}{\kappa_t(v)} \right]$, (Eq. (7): normalized by the capacity $\kappa_t(v)$)
  end for
  $h^{(r+1)} \leftarrow \mathrm{MPNN}(\tilde{h}^{(r)}, \chi_t)$ (Eq. (8))
  $s_i^{(r+1)}(k) \leftarrow s_i^{(r)}(k) + \eta \tanh\big(f_{\mathrm{score}}(h^{(r+1)}(\hat{p}_i^k))\big)$ (Eq. (9))
  $q_i^{(r+1)} \leftarrow \mathrm{softmax}(s_i^{(r+1)})$
end for
return $\{s_i^{(R)}\}_{i \in \mathcal{M}_t}$

Algorithm 4 Plan selection and execution (detail of Section 4.3)

Input: candidate plans $\{\hat{p}_i^k\}$ and candidate plan scores $\{s_i^{(R)}\}$ for all $i \in \mathcal{M}_t$; selection temperature τ
Output: executed joint action $\mathbf{a}_t$
for each agent $i \in \mathcal{M}_t$ (in parallel) **do**
  $z_i \sim \mathrm{softmax}(s_i^{(R)}/\tau)$
  $a_{i,t} \leftarrow \hat{p}_i^{z_i}[0]$
end for
$\mathbf{a}_t \leftarrow \{a_{i,t}\}_{i \in \mathcal{M}_t}$
submit $\mathbf{a}_t$ to the environment, update the state, observation, and cost (Section 4.3)
return $\mathbf{a}_t$