PLATO: Pointer Learner for Agent and Task Openness

Alireza Saleh Abadi
School of Computing
University of Nebraska-Lincoln
asalehabadi2@huskers.unl.edu

Leen-Kiat Soh
School of Computing
University of Nebraska-Lincoln
lksoh@unl.edu

Daniel Alan Redder
School of Computing
University of Georgia
daniel.redder@uga.edu

Adam Eck
Department of Computer Science
Oberlin College
aeck@oberlin.edu

Prashant Doshi
Institute for AI & School of Computing
University of Georgia
pdoshi@uga.edu

Abstract

Open agent systems (OASYS) are pervasive in the real world. In these systems, the sets of agents and tasks change over time. Such agent openness (AO) and task openness (TO) pose a fundamental challenge to multi-agent reinforcement learning, which typically assumes fixed state and action spaces. Existing methods address openness partially: padding and masking approaches introduce artificial bounds, while recent graph- or hypergraph-based methods handle one type of openness and make restrictive assumptions. In this paper, we introduce a pointer network-based actor combined with a centralized graph neural network (GNN) critic, trained with multi-agent proximal policy optimization under a centralized training and decentralized execution schema. In this method, which we refer to as Pointer Learner for Agent and Task Openness (PLATO), the pointer-based actor yields distributions directly over the current task set. This directly supports changing action spaces without masking or retraining. Our GNN critic encodes agent–task interactions as a graph that changes its topology with task and agent compositions. Together, these components reason about AO and TO while avoiding the boundedness of existing approaches. We formalize PLATO using a task- and agent-open Markov game (TaAgO-MG), generalizing prior task-open formalisms, and we prove that it is well defined over the resulting unbounded state and action spaces. We evaluate PLATO on the wildfire suppression domain available in the testbed Methods for Open Agent Systems Evaluation Initiative (MOASEI), a set of environments designed for open multi-agent system evaluation. We demonstrate strong performance and more consistent zero-shot generalization than state-of-the-art baselines for OASYS.

1 Introduction

In many real-world multi-agent systems, the world does not remain fixed while agents operate. New tasks may emerge unpredictably, agents may leave and join mid-operation, and their capabilities and

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

goals can change. Consider a wildfire suppression scenario where fires ignite, are suppressed or burn out, spread across the landscape, and where firefighters join and leave as their suppressant runs out. This dynamic property of the system, known broadly as openness [Eck et al., 2023], runs counter to the common assumptions of multi-agent reinforcement learning (MARL), where the sets of agents and tasks are assumed to be fixed during training and execution. Systems with these characteristics are referred to as *open* agent systems (OASYS). Two forms of openness commonly observed are agent openness (AO), where agents enter or leave, and task openness (TO), where tasks appear or disappear [Eck et al., 2023].

Handling openness is critical for robust deployment of MARL in practical settings, where a system must keep operating as its sets of agents and tasks change during execution. The challenge is that openness causes the underlying decision problem to change during execution: agents or tasks may appear or disappear, and policies that were optimal moments earlier may no longer apply. Environments are therefore *unbounded* in the quantity and composition of agents and tasks for which agents must coordinate; here *unbounded* means these counts cannot be fixed in advance, a statement about adaptability to changing composition rather than about performance at large scale. Practical MARL methods must adapt their outcomes to these changes without significant retraining.

Few MARL approaches address some facets of openness in multi-agent systems, but none address AO and TO jointly without limitations. For example, MOHITO [Anil et al., 2025] explicitly targets *task openness* as modeled by a task-open Markov game (TaO-MG). While its interaction hypergraph is sufficiently expressive to represent AO, MOHITO does not explicitly model or evaluate handling stochastic AO simultaneously. Deep implicit coordination graphs (DICG) [Li et al., 2021], though not explicitly designed for openness, introduce flexible coordination structures that provide limited support for AO and TO but still operate within bounded and padded inputs. Graph convolutional reinforcement learning (DGN) [Jiang et al., 2018] enables coordination by graph convolutions over an agent graph and can, in principle, accommodate changing agent neighborhoods through agent-centric graph structures. But it is not designed to model task openness and assumes a fixed action set. Deep Sets [Zaheer et al., 2017], a permutation-invariant set encoder, can summarize variable-sized task sets but decodes through a fixed-index action head that assumes a bounded, padded action set, and its pooled summary loses accuracy on set sizes and distributions unseen during training. Taken together, these methods represent important progress, yet none jointly addresses both AO and TO without restrictive assumptions or fixed bounds on the agent–task space.

We address these challenges with a novel actor-critic MARL algorithm named **Pointer Learner for Agent and Task Openness** (**PLATO**) that engages in centralized training and decentralized execution (CTDE) [Lowe et al., 2017]. PLATO uses a pointer network-based [Vinyals et al., 2015] actor, a critic that uses a centralized graph neural network (GNN) [Scarselli et al., 2008], and we train it using multi-agent proximal policy optimization (MAPPO) [Yu et al., 2022]. PLATO reasons about both TO and AO by learning a content-based pointing policy as introduced in pointer networks [Vinyals et al., 2015]. Agents point to tasks via query–key attention over the currently available tasks only, so the action distribution adapts as agents or tasks enter and leave without the need to retrain or bound the action and task sets.

The main contributions of this paper are: (1) **PLATO**, the first MARL architecture to handle both AO and TO jointly via content-based pointing, without fixed bounds or retraining; (2) a **TaAgO-MG** formalization extending prior task-open formulations, with formal guarantees that PLATO is permutation invariant and well defined over the resulting unbounded state and action spaces (Appendix A.1 and A.2, respectively); (3) an **empirical evaluation** of PLATO's performance and action efficiency across AO and TO setups, and grid sizes in the wildfire domain, comparing against smart and heuristic baselines; and (4) a **zero-shot generalizability assessment** by deploying trained policies on unseen grid sizes without retraining. Overall, PLATO represents a significant advance in robust learning-based methods that perform well in open agent systems.

2 Background

Open agent systems were first referenced more than two decades ago, but were not explicitly addressed until recently. We review OASYS below in Section 2.1 and a recent formalism for modeling decision making under task openness in Section 2.2. This is followed by some background on pointer networks that are suited for learning in OASYS.

2.1 Open Agent Systems (OASYS)

Openness in multi-agent systems refers to environments where either agents or tasks can change over time. A recent survey [Eck et al., 2023] and earlier discussions [Shehory, 2001, Calmet et al., 2004, Jumadinova et al., 2013] identify three main forms: (1) *Agent openness (AO)* occurs when agents join or leave the system, altering team composition and available capabilities; (2) *Task openness (TO)* arises when tasks appear or disappear during operation, requiring agents to adapt to new or vanishing opportunities; and (3) *Frame openness (FO)* refers to changes in agents' internal configurations–e.g., their capabilities and preferences, redefining how agents perceive and interact in the environment.

Among the three categories, AO has received the most attention in prior work. It was first defined over two decades ago as the addition of agents beyond the initial set [Shehory, 2001], and later refined as the complexity of transforming a system to add or remove an agent [Jamroga et al., 2013]. Subsequent studies showed that explicitly reasoning about the presence and absence of agents improves the behavior of the system [Chandrasekaran et al., 2016, Cohen et al., 2017, Kakarlapudi et al., 2022, Eck et al., 2023].

In comparison, TO is less studied and joint AO+TO settings remain unaddressed in a single learning framework. Some approaches model task arrivals and departures alongside agent dynamics [Jumadinova et al., 2013, Chen et al., 2016], and in MARL, MOHITO [Anil et al., 2025] explicitly addresses TO. Many domains exhibit exogenous TO while the agent set can also change online, requiring policies that remain well defined under both forms of openness.

While FO is about deeper capability and cognitive variability, AO and TO make the environment structure itself dynamic. Consequently, our work focuses on AO and TO, which directly determine the system's composition and coordination patterns and leaves FO for future work.

2.2 Task-Open Markov Games

Task-open Markov games model stochastic multi-agent decision-making problems, extending traditional Markov games [Littman, 1994] to task openness [Anil et al., 2025]. A TaO-MG is defined as:

$$\text{TaO-MG} \triangleq \langle M, X, \Psi \rangle$$

where • M, the base model, is a Markov game, in which N is the set of agents, S the state space, A_i the action space of agent i, $T : S \times \prod_{i \in N} A_i \to \Delta(S)$ the joint transition function, $R_i : S \times \prod_{i \in N} A_i \to \mathbb{R}$ the reward function for agent i, $\gamma \in [0, 1)$ the discount factor, and s_0 is the initial state. • X is the multiset of current tasks; • Ψ is a generator function whose components, $(\Psi_S, \Psi_A, \Psi_T, \Psi_R)$ for a Markov game, adapt M^t to M^{t+1} for a new set of tasks, X^{t+1}, at time $t + 1$.

Reinforcement learning in a TaO-MG constitutes each agent $i \in N$ learning a policy π_i that maximizes that agent's expected sum of discounted rewards,

$$\mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R_i^t(s^t, \mathbf{a}^t | s = s_0) \right],$$

where the trajectories are in the context of the current model, $\tau = \langle s^t, \mathbf{a}^t, X^t \rangle$, and $R_i^t = \Psi_R(R_i^{t-1}, X^t)$. This formalization makes task-driven changes explicit, but it leaves open how policies should act over a variable task set (and does not capture agent openness), motivating our TaAgO-MG extension and set-conditioned action selection in Section 3.

2.3 Pointer Networks

Pointer networks [Vinyals et al., 2015] generate outputs by selecting from the *current* input set rather than from a fixed vocabulary, making them naturally suited to variable-sized action spaces. This is directly relevant for OASYS: as tasks appear and disappear (TO), the available action set changes; as agents enter and leave (AO), the team context conditioning those selections changes. PLATO uses a single-step pointer at each timestep–each agent forms a query summarizing its team context and scores each available task via additive attention [Bahdanau et al., 2014]–yielding a distribution whose support matches the current task set. The formal mechanism is detailed in Section 4.1.

3 Agent and Task Openness in OASYS

We define the **task- and agent-open Markov game (TaAgO-MG)**, a generalization of TaO-MG (see Section 2.2) to consider both AO and TO. Since both the task and agent sets may change over time, the feasibility and utility of each task-action depend on the current tasks and team composition. Thus, a practical policy should evaluate the currently available tasks and the current team context, rather than assume a fixed, immutable set of actions.

In the following, we instantiate the Markov game using the popular wildfire suppression domain [Chandrasekaran et al., 2016, Kakarlapudi et al., 2022, Anil et al., 2025, Eck et al., 2020, Patino et al., 2025] that exhibits both TO and AO. Briefly, the environment is a grid where firefighter agents are stationed in fixed positions and each fire is a task with a location and size [Patino et al., 2025]. Fires may ignite (e.g., from lightning or spread), burn out, or be extinguished by firefighters, changing the set of active tasks and creating *task openness*. At each timestep, an agent chooses to either fight a fire or remain idle (NO-OP); it may also exit the environment when it is out of suppressant, creating *agent openness*. Rewards also vary: larger fires are more challenging to put out and rewarding, new fires create suppression opportunities, while burnouts and spread alter potential gains and losses.

Consider:

$$\text{TaAgO-MG} \triangleq \langle M, X, N, \Psi \rangle,$$

where X and N are the changing sets of tasks and agents, respectively, instead of a fixed set. We extend the generator Ψ from MOHITO [Anil et al., 2025] to also include N. Specifically, we treat Ψ as a generator that takes the base game M and the current open sets (X^t, N^t) and returns the time-indexed game, i.e., $M^t := \Psi(M, X^t, N^t)$. This lets us account for arbitrary deterministic change that new agents bring to the game. Formally, the generator is defined as

$$\Psi = \big\langle \Psi_S(M, X^t, N^t), \Psi_A(M, X^t, N^t), \Psi_T(M, X^t, N^t), \Psi_R(M, X^t, N^t) \big\rangle.$$

Ψ_S updates the state, Ψ_A updates the action set, Ψ_T generates the revised transition function, and Ψ_R generates the revised reward functions. This decomposition allows us to update all the components when the task set or agent set changes. Now, let A_i^t denote the action space of agent i at time t, and let $\mathsf{suppress}(x)$ denote the action of suppressing fire $x \in X^t$. (Note that in our wildfire suppression instantiation, each task has one action of suppressing that fire, so selecting a task uniquely determines the corresponding action.)

Task Arrival. If a new fire appears at time t, $X^{t+1} = X^t \cup \{x_{\text{new}}\}$. The generator induces the update of the action set, $A_i^{t+1} = A_i^t \cup \{\mathsf{suppress}(x_{\text{new}})\}, \ \forall i \in N^t$, corresponding to $\Psi_A(M, X^{t+1}, N^t)$.

Agent Departure. If agent i depletes its suppressant and exits, $N^{t+1} = N^t \setminus \{i\}$. The generator produces the updated game components: $S^{t+1} = S^t \setminus \{s_i\}$ and $R^{t+1} := \{R_j^t\}_{j \in N^{t+1}}$, which correspond to $\Psi_S(M, X^{t+1}, N^{t+1})$ and $\Psi_R(M, X^{t+1}, N^{t+1})$, respectively.

4 Methodology

PLATO is a CTDE method for multi-agent systems with TO and AO, whose architecture is shown in Figure 1. PLATO's core novelty is reformulating action selection from fixed-index classification to *content-based pointing*: each agent forms a context vector (query) and selects among the currently available tasks by matching this query against task feature vectors (keys). The policy output size is determined by the current task set, so tasks and agents can appear or disappear without retraining or architectural changes, keeping the policy well defined as openness changes the environment.

4.1 Pointer Actor

The pointer actor is the decentralized policy used at execution time. It supports TO by producing a policy whose support matches the currently available tasks, and supports AO by conditioning decisions on a permutation-invariant summary of the current team. The actor (1) deterministically factors o_i^t into an agent-feature matrix and a task-feature matrix, (2) encodes the current agent set into a fixed-dimensional query, (3) encodes each available task into a key, and (4) applies additive cross-attention to obtain a categorical distribution over the available task-actions.

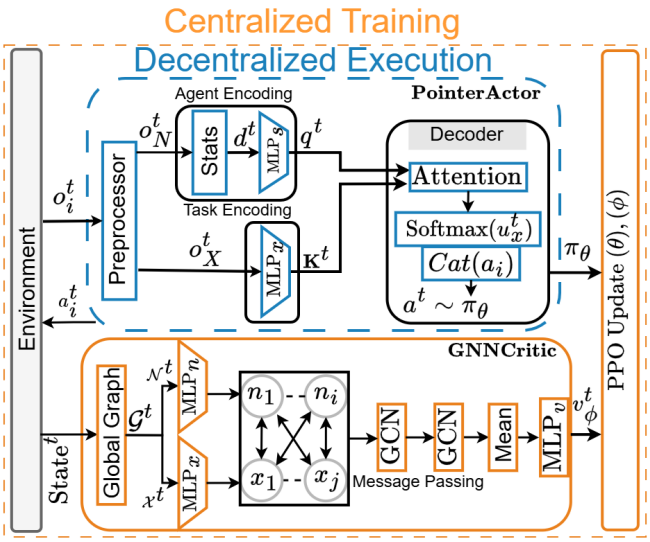

Figure 1: **PLATO architecture for OASYS.** PLATO operates under CTDE: orange components are used only during the centralized training, and blue components during the decentralized execution. The PointerActor scores available tasks by attending from a team-summary query to task keys (Section 4.1). The GNNCritic estimates the state value from a bipartite agent–task graph (Section 4.3). Both are padding-free and trained with MAPPO (Section 4.5).

4.2 Observation Factorization

In TaAgO-MG, the observation of the ith agent at time t, o_i^t, is its local view of the global state, State^t, restricted to the current agent set N^t and task set X^t that are visible or relevant to agent i. The preprocessor factors o_i into an agent-feature matrix $o_N^t \in \mathbb{R}^{|N^t| \times f_N}$ and a task-feature matrix $o_X^t \in \mathbb{R}^{|X^t| \times f_X}$. In these feature matrices, rows correspond to i's observation of tasks and other agents, similar to node factorization for MOHITO's critic graph [Anil et al., 2025].

Algorithm 1 PLATO: POINTER ACTOR, GNN CRITIC AND TRAINING

Inputs: environment Env, horizon T, actor parameters θ, critic parameters ϕ, initial observation vector O^0
for $t = 1 \dots T$ **do**
  $a_i^t \sim \text{POINTERACTOR}(o_i^t) \quad \forall i \in N^t$
  $O^t, \text{State}^t, r^t \leftarrow Env(a^t)$
  $\hat{V}^t \leftarrow \text{GNNCRITIC}(\text{State}^t; \phi)$; store $(\cdot)$
  compute GAE $\hat{A}^t$; update θ by PPO-clip; update ϕ by MSE$(\hat{V}^t, \hat{R}^t)$
end for

procedure POINTERACTOR(O_i^t)
  $o_N^t, o_X^t \leftarrow \text{PREPROCESSOR}(O_i^t)$
  $d^t \leftarrow [\mu, \text{Var}, \min, \max](o_N^t)$
  $q^t \leftarrow \text{MLP}_s(d^t)$
  $K^t \leftarrow \{\text{MLP}_x(o) \mid o \in o_X^t\} \cup \{\text{MLP}_x(o_{\text{NO-OP}})\}$
  $u_x^t \leftarrow \mathbf{v}^\top \tanh(\mathbf{W}_K k_x^t + \mathbf{W}_q q^t) \quad \forall k_x^t \in K^t$
  return $\text{Cat}(\text{softmax}([u_x^t]_{x \in X^t \cup \{\text{NO-OP}\}}))$
end procedure

procedure GNNCRITIC(State^t)
  form $\mathcal{G}^t = \langle \mathcal{X}^t, \mathcal{N}^t, \mathcal{E}^t \rangle$
  $h_0^t \leftarrow \text{Concat}\left[\text{MLP}_{\mathcal{N}}(\mathcal{N}^t), \text{MLP}_{\mathcal{X}}(\mathcal{X}^t)\right]$
  $h_l^t \leftarrow \text{GCN}(h_{l-1}^t, \mathcal{E}^t)$ for each layer l
  $M^{(L)} \leftarrow \frac{1}{|\mathcal{N}^t|+|\mathcal{X}^t|} \sum_i (h_L^t + h_0^t)_i$
  return $V = W_v \, \text{MLP}_v(M^{(L)}) + b_v$
end procedure

Preprocessor. Each agent i receives a local observation $o_i^t \in O^t$ covering the agents and tasks relevant to i at time t. A deterministic preprocessor factors this into an agent-feature matrix and a task-feature matrix:

$$o_N^t \in \mathbb{R}^{|N^t| \times f_N}, \quad o_X^t \in \mathbb{R}^{|X^t| \times f_X}.$$

Encoder. This step constructs one *query* from the agent set and a variable-sized set of *keys* from the task set. We use MLP encoders since the actor makes a single decision per timestep from fully observable features; sequential decoding is not required. We address AO by aggregating four statistics (i.e., mean, variance, minimum, and maximum) over agents in o_N^t to produce a fixed-

length vector d^t (Alg. 1, line 10) and then encoding it as $q^t = \mathrm{MLP}_s(d^t)$ (Alg. 1, line 11). These statistics make the encoder permutation invariant to agent order (Lemma 1, Section 4.4). We use multiple statistics to reduce aliasing, where different agent sets could otherwise produce identical representations. We address TO through attention. We construct $|X^t| + 1$ keys K^t from o_X^t (Alg. 1, line 12) in the task encoder: one key $k_x^t = \mathrm{MLP}_x(o_x^t)$ per task, plus a NO-OP key. We assume one action per task with a persistent NO-OP, so $a_i^t \in \{\text{NO-OP}\} \cup \{a_x \mid x \in X^t\}$.

Pointer Decoder. The Pointer Decoder is a one-step variant of a pointer network [Vinyals et al., 2015] that maps the *query* and *keys* to a stochastic policy over currently available *task-actions*. We use one-step, additive attention [Bahdanau et al., 2014] to decode (q^t, k_x^t) to u_x^t, a scalar attention score (Alg. 1, line 13), in which $\mathbf{W}_K$ and $\mathbf{W}_q$ are learnable projection matrices, and $\mathbf{v}$ is a learnable vector projecting the compatibility space to a scalar score. We then normalize over all attention scores using softmax to obtain the policy (Alg. 1, line 14).

4.3 GNN Critic

The single critic requires centralized training and estimates $V_\phi(s^t)$. It represents the open global state as a bipartite agent–task graph whose node sets vary with $|N^t|$ and $|X^t|$. Agent and task features are embedded into a shared latent space, message passing propagates relational information across agent–task edges, and pooling yields a fixed-size graph summary that is mapped to a scalar value. This avoids fixed-size concatenation and supports variable numbers of agents and tasks.

Graph $\mathcal{G}^t = \langle \mathcal{X}^t, \mathcal{N}^t, \mathcal{E}^t \rangle$ encodes the state, State^t, at time t, where (1) $\mathcal{X}^t \in \mathbb{R}^{|X^t| \times f_X}$ are *task nodes* and contain f_X many features, describing each task at time t; (2) $\mathcal{N}^t \in \mathbb{R}^{|N^t| \times f_N}$ are *agent nodes* and contain f_N many features, describing each agent at time t; and (3) $\mathcal{E}^t$ includes undirected edges between all agent–task pairs which have an action, $\{\langle n, x \rangle | n \in \mathcal{N}^t \text{ and } x \in \mathcal{X}^t\}$.

We build a critic network using a graph convolutional network (GCN) [Kipf and Welling, 2016] to propagate information between agent and task nodes by message passing. First, we use MLP encoders to map $\mathcal{N}^t$ and $\mathcal{X}^t$ into a shared latent space and concatenate them to form $\mathbf{h}_0^t$ (Alg. 1, line 18), where $\mathbf{h}_0^t$ is the embedding of all nodes in $\mathcal{G}^t$ and f_h is the hidden dimension. We then perform message passing (Alg. 1, line 19) with a skip connection [He et al., 2016] that adds $\mathbf{h}_0^t$ to the final layer output before pooling, preserving individual node features that would otherwise be smoothed out by message passing. We use two layers to allow information to propagate between agents and tasks through the bipartite graph. Finally, we map the node embeddings to a scalar value by mean pooling over nodes followed by an MLP_v value head (Alg. 1, lines 20–21).

4.4 Theoretical Properties

Under openness the agent and task sets vary over time, yet PLATO operates on this unbounded space of states with a single fixed set of parameters. The actor and critic are *well defined* if each maps every state to exactly one output – a single action distribution for the actor and a single value for the critic, regardless of how many agents and tasks are present or how they are ordered. We state this in Theorem 1 and prove it in Appendix A.2.

Lemma 1 (Encoder Permutation Invariance)**.** *Each team statistic is permutation invariant,* MLP_s *operates on the aggregate, and* MLP_x *operates on task observations independently. Therefore, the agent and task encodings are permutation invariant.*

The proof for this lemma is in Appendix A.1.

Theorem 1 (PLATO is well defined)**.** *The actor and critic are permutation invariant per Lemma 1 and Appendix A.2. The MLPs act on observations independently or on aggregates. The pointer decoder and critic GCNs are defined over a single unbounded dimension. Therefore, the actor and critic are well-defined functions over the countably infinite open agent–task space.*

Proof sketch. The actor and critic satisfy existence and uniqueness. The fixed-length statistics $d^t \in \mathbb{R}^{4f_N}$, per-task keys, and graph nodes feed maps whose input size is independent of $|N^t|$ and $|X^t|$, so the softmax and pooled value head are always defined (existence); as these maps are deterministic and permutation invariant (Lemma 1), the output is unique (uniqueness). See Appendix A.2 for the full proof. □

Theorem 2 (Complexity Analysis for PLATO)**.** *Treating all neural network dimensions and the number of GCN layers as constants, the per-timestep cost of one PLATO training step, combining the actor forward pass over all $|N^t|$ agents with the critic forward pass, is $\mathcal{O}(|N^t|^2 + |N^t||X^t|)$.*

4.5 Training

We train PLATO under CTDE using MAPPO [Yu et al., 2022, Schulman et al., 2017] with a centralized critic and a single shared policy, while at execution time only the decentralized pointer actor is used. The actor is optimized with the PPO clipped objective [Schulman et al., 2017]. The critic uses the standard PPO clipped value regression loss, and we compute advantages with the generalized advantage estimation (GAE) [Schulman et al., 2015] (Alg. 1, line 6). All actor ($\theta = \{\mathrm{MLP}_S, \mathrm{MLP}_X, \mathbf{W}_K, \mathbf{W}_q\}$) and critic ($\phi = \{\mathrm{MLP}_{\mathcal{X}}, \mathrm{MLP}_{\mathcal{N}}, MLP_v, \mathrm{GCNs}\}$) components are standard differentiable modules optimized jointly. The per-timestep training cost holds all neural network dimensions constant and scales only with the number of active agents $|N^t|$ and active tasks $|X^t|$, analyzed in Section 4.4 (Theorem 2, proof in Appendix A.5).

5 Experiments

We evaluate PLATO, available at *https://github.com/oasys-mas/PLATO*, against four sophisticated baselines and four naive heuristic-based baselines across TO and AO configurations on 2×3, 3×3, and 4×4 grids in our experiments. Note that we distinguish *native* evaluation (training and testing on the same grid) from *zero-shot* evaluation (deploying a trained policy on an unseen grid *without* retraining); see Appendix C.2 and C.2.1 for setup illustrations and zero-shot details. Also, additional native and zero-shot results are in Appendix C.7.

Domain and Setups. We use the WILDFIRE domain [Patino et al., 2025] (Appendix C.1): an $r \times c$ grid where firefighter agents at fixed positions suppress fires given finite suppressant. *Endogenous TO* arises from fire *spread* (existing fires probabilistically ignite neighboring cells); *exogenous TO* from *random ignitions*; and *AO* from stochastic suppressant depletion triggering an agent's exit-and-refill cycle. At each step, agents may suppress a fire in their range or select NO-OP. The extinguishment of a fire of intensity s yields $+2^s$ while a burnout costs -2^s to an agent. We define four setups (S0–S3) over a fixed 100-step horizon (Table 6, Appendix C.3): S0 is a *closed* baseline (no spread, no ignitions, unlimited suppressant); S1 adds *endogenous TO* (spread prob. = 0.6, refill prob. = 1.0); S2 introduces *exogenous TO* (ignition prob. = 0.6, refill prob. = 0.8); S3 combines *both S1 and S2 with refill prob.* = 0.6, the most demanding joint AO+TO condition. Openness event counts and grid configuration details are in Appendix C.4 and C.5.

Models and Baselines. Our method is **PLATO**, as defined in Algorithm 1. The four sophisticated baselines are: **DGN** [Jiang et al., 2018] and **DICG** [Li et al., 2021] represent state-of-the-art MARL methods for dynamic coordination but assume a fixed set of agents and tasks during training; **MO-HITO** [Anil et al., 2025] is the closest prior method explicitly designed for TO; and **DeepSets** [Zaheer et al., 2017] is a permutation-invariant set encoder that, like PLATO, ingests variable-sized agent and task sets, but scores tasks from a pooled set embedding through a fixed-index head rather than by content-based pointing. We also use four naive baselines to contextualize gains from coordination versus simple rule-based behavior: **NOOP** always selects NO-OP, providing a lower bound; **Random** samples uniformly from available actions; **Strongest** prioritizes the highest-intensity fires, representing an urgency-driven heuristic; and **Weakest** prioritizes the lowest-intensity fires, reflecting an immediacy-driven strategy.

Metrics. We evaluate policy performance using **episode return** (average cumulative reward per episode) and policy efficiency using **reward per fight action**. Additionally, we report on domain-specific metrics such as *burnouts*, *putouts*, and NOOP usage by the firefighters for each test in Appendix C. Charts report mean$\pm$std across 150 evaluation episodes (50 seeds per checkpoint, 3 checkpoints per method). We determine significance with a Shapiro-Wilk-adaptive two-sided test [Wilcoxon, 1945, Shapiro and Wilk, 1965] (two-sided paired t-test or Wilcoxon signed-rank, selected by Shapiro-Wilk normality pre-test at the same Bonferroni-corrected α; see Appendix C.6), with Bonferroni correction [Hochberg and Tamhane, 1987] ($\alpha = 0.05/\text{methods}$). We report the best results for each learned policy from Optuna hyperparameter tuning [Akiba et al., 2019]. Details on tuning, seeding, and compute resources are in Appendix B and C.6 (Table 10).

Model	trained and tested on 3×3				trained and tested on 4×4			
	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	8.00±0.00	7.52±1.46	**112.83±12.62**	**80.53±19.66**	12.00±0.00	**19.47±9.33**	**228.81±21.99**	**149.35±27.33**
DICG	8.00±0.00	5.97±2.53	59.75±13.69	40.83±15.90	12.00±0.00	12.11±3.95	128.63±21.97	74.39±23.54
DGN	7.63±1.17	5.52±2.81	44.59±15.02	31.56±16.62	11.79±0.90	10.83±4.83	82.99±20.57	64.16±19.27
MOHITO	8.00±0.00	3.04±3.64	42.36±19.89	47.24±18.24	12.00±0.00	6.11±6.36	41.97±28.71	33.91±19.85
DeepSets	8.00±0.00	6.61±2.32	80.55±20.75	59.36±17.12	12.00±0.00	13.76±6.57	165.93±22.92	111.67±26.78
NOOP	−8.00±0.00	−8.00±0.00	−12.00±0.00	−12.00±0.00	−12.00±0.00	−12.00±0.00	−16.00±0.00	−18.00±0.00
Random	6.00±2.46	4.48±3.49	11.48±16.06	0.88±12.96	9.20±3.16	3.20±4.44	49.76±21.11	35.52±22.49
Weakest	8.00±0.00	7.76±0.96	70.36±22.66	37.44±1.72	12.00±0.00	10.36±3.12	137.36±20.15	49.40±0.93
Strongest	8.00±0.00	7.28±1.75	24.24±29.13	13.92±20.38	12.00±0.00	7.68±4.75	161.24±12.37	80.64±11.78

Table 1: Performance: Episode return (mean ± std). Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

Model	trained and tested on 3×3				trained and tested on 4×4			
	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	**1.25±0.23**	0.60±0.38	**0.74±0.08**	**0.64±0.10**	1.09±0.25	0.41±0.21	**0.73±0.06**	**0.66±0.07**
DICG	0.73±0.39	0.35±0.21	0.45±0.09	0.37±0.16	0.71±0.35	0.56±0.22	0.51±0.07	0.41±0.11
DGN	0.61±0.41	0.48±0.46	0.40±0.11	0.31±0.19	0.44±0.34	0.42±0.23	0.43±0.08	0.39±0.13
MOHITO	0.97±0.12	0.25±0.39	0.23±0.59	0.39±0.16	**1.21±0.10**	0.27±0.28	0.25±0.33	0.24±0.33
DeepSets	1.08±0.20	0.44±0.25	0.59±0.20	0.58±0.10	1.07±0.22	0.48±0.27	0.59±0.07	0.51±0.12
NOOP	–	–	–	–	–	–	–	–
Random	0.73±0.36	0.38±0.39	0.03±0.49	−0.22±0.47	0.81±0.38	0.12±0.19	0.36±0.22	0.29±0.20
Weakest	1.03±0.13	**0.87±0.44**	0.48±0.07	0.35±0.02	1.07±0.32	0.35±0.17	0.54±0.04	0.42±0.01
Strongest	1.03±0.13	0.71±0.57	−0.10±0.62	−0.16±0.57	1.07±0.32	0.18±0.12	0.53±0.04	0.41±0.03

Table 2: Efficiency: Reward per fight (mean ± std). Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$). NOOP takes no fight actions, so its reward per fight is undefined (–).

6 Results

Performance. Table 1 shows (1) PLATO achieves the highest return in S2 and S3 on the 3×3 grid with statistical significance, while performing as well as top baselines in S1, and (2) PLATO outperforms all baselines in S1–S3 with statistical significance on the 4×4 grid. Furthermore, in S0 (no openness), all smart methods except DGN match. DeepSets is the strongest sophisticated baseline, with performance only second to PLATO among learned methods in S1–S3 on the 3×3 and 4×4 grids, yet PLATO's advantage over it stays significant in S2–S3 on both grids. Results on the 2×3 grid are in Appendix C.7.

Efficiency. Table 2 shows PLATO attains the highest reward per fight–i.e., policy efficiency–on the 3×3 grid in S0, S2, and S3 with statistical significance while Weakest is the best in S1. On the 4×4 grid, PLATO leads S2–S3 and MOHITO leads S0, with statistical significance. PLATO's performance remains positive across all grids and setups. Results on the 2×3 grid are in Appendix C.7.

Generalizability. Here we empirically evaluate PLATO's performance in zero-shot generalization by deploying learned policies on a different unseen grid without retraining, where the same team faces different fires and novel positions (see Appendix C.2.1). Table 3 reports the zero-shot episode returns for both training sizes. When trained on 2×3, (1) and applied to 3×3: PLATO leads S1–S2 with statistical significance, (2) and applied to 4×4: DICG leads S1 while PLATO and DeepSets are statistically tied (no significant best) in S2 and MOHITO collapses to negative returns in S2. When trained on 3×3, (1) and applied to 5×5: PLATO leads S1–S3 with statistical significance, and (2) and applied to 4×4: DICG leads S1. Also, DGN is among the two weakest learned methods in most open zero-shot settings. Additional zero-shot results are in Appendix C.7.2.

Ablation. For our ablation studies, we test *four* variants from a factorial design crossing two encoder choices (MLP, LSTM) with two scorer choices (additive, dot-product): **(mlp+add)**, **(mlp+dot)**, **(lstm+add)**, and **(lstm+dot)**. Table 4 reports the native and zero-shot episode-return results. Additional metrics are in Appendix C.8. Our analysis is driven by questions such as *Q1: Does tem-*

	trained on 2×3 and tested (zero-shot) on 3×3				trained on 2×3 and tested (zero-shot) on 4×4			
Model	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	9.97±0.33	**6.83±2.87**	**80.75±12.65**	4.95±28.12	11.73±2.51	5.31±9.89	64.11±15.58	13.68±36.62
DICG	10.00±0.00	3.44±3.15	60.63±15.37	25.31±18.21	11.27±1.97	**8.17±10.17**	59.04±16.85	48.69±18.48
DGN	7.07±2.77	2.91±2.46	51.84±12.29	27.16±18.51	4.11±3.64	4.23±8.86	39.61±25.76	44.88±21.18
MOHITO	3.87±2.88	0.96±3.12	24.39±26.92	−4.63±13.48	3.41±2.08	−1.65±4.33	−13.17±9.09	23.24±22.84
DeepSets	8.56±2.19	5.28±3.56	70.15±17.49	22.47±26.03	7.65±3.41	0.44±5.05	61.12±19.69	49.72±24.19
	trained on 3×3 and tested (zero-shot) on 4×4				trained on 3×3 and tested (zero-shot) on 5×5			
Model	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	9.65±2.70	5.23±9.58	49.17±25.38	57.36±14.29	7.76±2.98	**2.48±6.17**	**80.79±21.55**	**50.40±26.18**
DICG	10.00±1.47	**8.13±8.42**	61.88±13.04	51.36±16.28	9.23±1.85	0.07±4.47	41.51±18.04	21.51±19.15
DGN	6.80±3.88	−0.57±5.45	47.85±19.51	35.93±26.71	6.69±3.78	−2.80±4.19	28.29±14.36	14.21±18.18
MOHITO	11.20±1.84	−1.72±6.56	34.68±30.94	55.77±15.25	9.52±1.95	−0.43±4.14	40.79±22.12	25.36±21.85
DeepSets	10.80±2.91	1.48±7.63	62.24±12.28	59.16±14.92	7.84±2.68	−0.40±4.64	56.92±22.19	33.84±22.62

Table 3: Generalizability: Episode return (mean ± std), zero-shot transfer. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.01$).

	trained and tested on 3×3				trained and tested on 4×4			
PLATO	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	8.00±0.00	7.52±1.46	112.83±12.62	80.53±19.66	12.00±0.00	19.47±9.33	228.81±21.99	149.35±27.33
(mlp+dot)	8.00±0.00	7.28±1.54	109.65±13.60	84.37±20.79	12.00±0.00	19.76±9.51	221.04±24.03	146.63±25.99
(lstm+add)	8.00±0.00	7.47±1.36	108.56±14.33	82.12±22.23	12.00±0.00	16.12±6.75	227.59±24.53	151.33±24.35
(lstm+dot)	8.00±0.00	7.20±1.61	113.67±15.30	88.16±17.87	12.00±0.00	19.49±8.57	218.79±23.32	148.31±26.50
	trained on 2×3 and tested (zero-shot) on 3×3				trained on 2×3 and tested (zero-shot) on 4×4			
PLATO	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	9.97±0.33	6.83±2.87	80.75±12.65	4.95±28.12	11.73±2.51	5.31±9.89	64.11±15.58	13.68±36.62
(mlp+dot)	10.00±0.00	6.25±3.01	78.19±12.02	4.52±27.48	**12.96±1.93**	4.09±8.57	63.17±16.43	17.89±36.82
(lstm+add)	9.84±0.91	6.52±2.64	71.79±22.24	**18.49±27.48**	11.09±2.78	7.37±9.37	66.88±11.67	**37.97±32.54**
(lstm+dot)	9.89±0.80	6.31±3.39	82.87±16.74	6.19±28.53	10.32±2.95	4.56±6.30	65.76±18.18	25.56±36.38

Table 4: Ablation: Episode return (mean ± std). *Top:* native evaluation; no variant achieves statistically significant best performance. *Bottom:* zero-shot transfer. Bold indicates statistically significant best performance. Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$.

poral memory in the encoder improve performance? We compare **(mlp+add) vs. (lstm+add)** and **(mlp+dot) vs. (lstm+dot)** to isolate the encoder across both scoring functions. In native evaluation, no LSTM or MLP variant is statistically best under either scorer. In zero-shot transfer, the encoder's impact depends on the scorer and setup. With additive scoring, (lstm+add) achieves significantly higher returns than (mlp+add) in S3 across both target grids; (lstm+dot) is also numerically higher than (mlp+dot) in S3 but neither is statistically best. Together, these results show that the benefit of temporal memory due to LSTM is not universal: it is most pronounced under combined AO+TO (S3) in zero-shot settings when paired with additive scoring. *Q2: How does additive attention scoring perform with respect to dot-product scoring?* We compare **(mlp+add) vs. (mlp+dot)** and **(lstm+add) vs. (lstm+dot)**. In native evaluation, no variant is statistically best under either encoder. In zero-shot transfer, the scorer's advantage also depends on the encoder and setup. With the LSTM encoder, (lstm+add) achieves statistically significantly higher returns than (lstm+dot) in S3 across both target grids – additive scoring wins in the most open zero-shot setting. With the MLP encoder, (mlp+dot) achieves statistically significantly higher returns than (mlp+add) in S0 at 4×4 – dot-product scoring wins in the simplest zero-shot setting. Together with *Q1*, these results point to a consistent interaction: *(lstm+add) is the best combination under combined AO+TO (S3), while (mlp+dot) is the best combination under the closed environment (S0) in zero-shot transfer.*

7 Related Work

Graph neural networks (GNNs) [Scarselli et al., 2008] model multi-agent interactions flexibly by representing agents as graph nodes and using message passing to aggregate neighbor information, naturally handling variable group sizes but often limiting openness by imposing bounded-

ness assumptions. Several dynamic multi-agent approaches partially address related challenges. DGN [Jiang et al., 2018] uses graph convolution with fixed-size neighbor communication for cooperation and can generalize to larger teams but restricts interaction through a predefined neighborhood and assumes static goals. DICG [Li et al., 2021] learns dynamic coordination graphs for multi-agent coordination, handling large fixed teams well but requires known agent and task sets in advance, limiting true openness. Graph-based policy learning (GPL) [Rahman et al., 2021] tackles AO in ad hoc teamwork with graph neural networks and type inference for variable team sizes but trains only a single agent while others have fixed policies, thus not a fully adaptive multi-agent training approach. In contrast, MOHITO [Anil et al., 2025] is specifically designed for openness, formalizing task-open Markov games and employing hypergraph-based actor-critics to handle dynamic task-action sets, though it focuses solely on TO and assumes fixed agents and frames. PLATO achieves strong performance under combined openness (S2–S3) and more consistent zero-shot generalization than these baselines, with advantages that widen at larger grid scales.

Although transformers [Vaswani et al., 2017] theoretically support variable-length inputs by attending over all tokens (making them attractive to OASYS), bounds are needed for training efficiency: to enable parallelization, *inputs are typically padded to fixed lengths*. This practice limits transformer-based applications in OASYS to a maximum number of agents or tasks. PLATO, on the other hand, circumvents this limitation via the attention mechanism and team-based statistics to handle variability in agents and tasks. We additionally compare against Deep Sets [Zaheer et al., 2017], a permutation-invariant set-encoder baseline that pools entity embeddings into a fixed-size representation, handling variable-sized agent and task sets without padding. Set Transformers [Lee et al., 2019] are a related permutation-invariant alternative, but add attention cost and, in practice, still pad to a maximum length. To the best of our knowledge, no existing transformer-based MARL method targets joint AO and TO, so such a comparison would require an openness-aware transformer of our own design, which we leave to future work.

Related fields such as multi-task reinforcement learning [Zhang and Yang, 2021] and curriculum learning [Narvekar et al., 2020] adapt to varying tasks or harder scenarios but assume a fixed entity count and do not address AO. They may also suffer catastrophic forgetting [Ribeiro et al., 2019], high prediction errors in volatile environments, and do not handle openness, relying on predefined task spaces or curriculum structures [Kravchenko and Cusack, 2024, Abadi and Soh, 2025].

8 Conclusion

Actors and tasks seldom stay the same over a period of time. We introduced **PLATO**, which combines (i) a pointer-network actor that produces action distributions specific to the current model of an open system and invariant to the order of agents and tasks in that system, and (ii) a central GNN critic which interprets a novel bipartite agent–task state representation for open games with one action per task. This design enables policies to succinctly encode actions over a large dynamic space. It does this without padding-based bounds or retraining. We formalized the task and agent open setting with TaAgO-MG and proved that PLATO is well-defined and permutation-invariant over TaAgO-MG's unbounded state and action spaces (Appendix A.2).

PLATO demonstrates significant improvement over padded MARL and prior open methods on many of our wildfire experiments. These results, combined with significant zero-shot generalization, show PLATO learns a robust policy that reasons about openness. Our ablations revealed an unexpected behavior. Excluding temporal memory improved our zero-shot results in simple settings.

This study is limited. Our evaluation is confined to a single domain. Wildfire is one of the few established testbeds exhibiting AO and both forms of TO. Many state-of-the-art MARL techniques are incompatible with dynamic spaces. We compared against the strongest permutation-invariant baselines, though they are not transformers. How well a transformer-based MARL technique would compare is an open question. The largest system we evaluate has six agents and at most a few dozen tasks. PLATO's behavior with fifty or more tasks is unknown. We suspect ambivalence between similar tasks and a diffuse softmax would especially complicate learning with many present tasks.

Future work could involve loosening assumptions about the environment and a further study of temporal memory under openness. Partial observability and agent type openness present challenging relaxations to consider. These directions will bring PLATO closer to reliable and performant policies for real-world systems.

Acknowledgments and Disclosure of Funding

This research was supported, in part, by a collaborative NSF grant #IIS-2312657 (to Doshi), #IIS-2312658 (to Soh), and #IIS-2312659 (to Eck), and NSF grant #IIS-2530002 to Doshi. Some of the computing occurred on the Holland Computing Center of the University of Nebraska, which receives support from the university's Office of Research and Economic Development and the Nebraska Research Initiative. We thank T. Billings and C. Patino IV for their input on the project, including support for the wildfire domain. We also thank the Area Chair and Reviewers who provided constructive feedback that improved the paper.

References

Alireza Saleh Abadi and Leen-Kiat Soh. Challenges in credit assignment for multi-agent reinforcement learning in open agent systems. *arXiv preprint arXiv:2510.27659*, 2025.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

Gayathri Anil, Prashant Doshi, Daniel Alan Redder, Adam Eck, and Leen-Kiat Soh. Mohito: Multi-agent reinforcement learning using hypergraphs for task-open systems. In *The 41st Conference on Uncertainty in Artificial Intelligence*, 2025.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

J. Calmet, A. Daemi, R. Endsuleit, and T. Mie. A liberal approach to openness in societies of agents. In *Engineering Societies in the Agents World IV, LNCS*, volume 3071, pages 81–92, 2004.

Muthukumaran Chandrasekaran, Adam Eck, Prashant Doshi, and Leenkiat Soh. Individual planning in open and typed agent systems. In *UAI'16*, pages 82–91. AUAI Press, 2016.

Bin Chen, Adam Eck, and Leen-Kiat Soh. Collaborative human task assignment for open systems. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 1441–1442, 2016.

Jonathan Cohen, Jilles-Steeve Dibangoye, and Abdel-Illah Mouaddib. Open decentralized pomdps. In *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 977–984. IEEE, 2017.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Adam Eck, Maulik Shah, Prashant Doshi, and Leen-Kiat Soh. Scalable decision-theoretic planning in open and typed multiagent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7127–7134, 2020.

Adam Eck, Leen-Kiat Soh, and Prashant Doshi. Decision making in open agent systems. *AI Magazine*, 44(4):508–523, December 2023. ISSN 0738-4602, 2371-9621. doi: 10.1002/aaai.12131. URL https://onlinelibrary.wiley.com/doi/10.1002/aaai.12131.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Yosef Hochberg and Ajit C. Tamhane. *Multiple Comparison Procedures*. John Wiley & Sons, New York, 1987.

W. Jamroga, A. Meski, and M. Szreter. Modularity and openness in modeling multi-agent systems. In *GandALF*, pages 224–239, 2013.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*, 2018.

Janyl Jumadinova, Prithviraj Dasgupta, and Leen-Kiat Soh. Strategic capability-learning for improved multiagent collaboration in ad hoc environments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(8):1003–1014, 2013.

Anirudh Kakarlapudi, Gayathri Anil, Adam Eck, Prashant Doshi, and Leen-Kiat Soh. Decision-theoretic planning with communication in open multiagent systems. In *UAI*, pages 938–948, 2022.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Anna Kravchenko and Rhodri Cusack. The limitations of automatically generated curricula for continual learning. *Plos one*, 19(4):e0290706, 2024.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

Sheng Li, Jayesh K. Gupta, Peter Morales, Ross Allen, and Mykel J. Kochenderfer. Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning, February 2021. URL http://arxiv.org/abs/2006.11438. arXiv:2006.11438 [cs].

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, pages 157–163. PMLR, 1994.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

George B. Mertzios, Hendrik Molter, Rolf Niedermeier, Viktor Zamaraev, and Philipp Zschoche. Computing maximum matchings in temporal graphs. *Journal of Computer and System Sciences*, 137:1–19, 2023. doi: 10.1016/j.jcss.2023.04.005.

Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, Hoboken, NJ, 9th edition, 2017.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Ceferino Patino, Tyler J Billings, Alireza Saleh Abadi, Daniel Redder, Adam Eck, Prashant Doshi, and Leen-Kiat Soh. Inaugural moasei competition at aamas'2025: A technical report. *arXiv preprint arXiv:2507.05469*, 2025.

Muhammad A. Rahman, Niklas Hopner, Filippos Christianos, and Stefano V. Albrecht. Towards open ad hoc teamwork using graph-based policy learning. In *International conference on machine learning*, pages 8776–8786. PMLR, 2021. ISBN 2640-3498.

Joao Ribeiro, Francisco S Melo, and Joao Dias. Multi-task learning and catastrophic forgetting in continual reinforcement learning. *arXiv preprint arXiv:1909.10008*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. ISBN: 1045-9227 Publisher: IEEE.

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.

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

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 1965. doi: 10.1093/biomet/52.3-4.591.

Onn Shehory. Software architecture attributes of multi-agent systems. In Paolo Ciancarini and Michael J. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 77–90, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44564-7.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987. URL http://www.jstor.org/stable/3001968.

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. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609, 2021.

A Proofs and Complexity

A.1 Lemma 1: Permutation Invariance of Team Statistics

Here we show that the set of statistics we use to approximate the state of present agents is permutation invariant.

Lemma 1 (Encoder Permutation Invariance)**.** *Each team statistic is permutation invariant, MLP_s operates on the aggregate, and MLP_x operates on task observations independently. Therefore, the agent and task encodings are permutation invariant.*

Proof. Let $o_N^t : [o_1^t, ...o_n^t] \in \mathbb{R}^{n \times f_N}$ be the agent-feature matrix for a team of $n = |N^t|$ agents, and let σ be any permutation of $\{1, \dots, n\}$. Then the team-statistics vector

$$d^t = \left[\mu(o_N^t),\ \mathrm{Var}(o_N^t),\ \min(o_N^t),\ \max(o_N^t)\right] \in \mathbb{R}^{4f_N}$$

is invariant to agent ordering $d^t\big(\sigma(o_N^t)\big) = d^t\big(o_N^t\big)$ if all statistics in d^t are permutation invariant.

Mean. Because σ is a bijection on $\{1, \dots, n\}$, the index set $\{\sigma(1), \dots, \sigma(n)\}$ equals $\{1, \dots, n\}$, so

$$\mu_j\big(\sigma(o_N^t)\big) = \frac{1}{n}\sum_{i=1}^{n} o_{\sigma(i),j}^t = \frac{1}{n}\sum_{i=1}^{n} o_{i,j}^t = \mu_j\big(o_N^t\big).$$

Variance. Since μ_j is invariant (above), the same bijection argument gives

$$\mathrm{Var}_j\big(\sigma(o_N^t)\big) = \frac{1}{n}\sum_{i=1}^{n}\big(o_{\sigma(i),j}^t - \mu_j\big)^2 = \frac{1}{n}\sum_{i=1}^{n}\big(o_{i,j}^t - \mu_j\big)^2 = \mathrm{Var}_j\big(o_N^t\big).$$

Min and max. The minimum (resp. maximum) of a finite multiset depends only on the set of values, not their order:

$$\min_i\ o_{\sigma(i),j}^t = \min_i\ o_{i,j}^t, \qquad \max_i\ o_{\sigma(i),j}^t = \max_i\ o_{i,j}^t.$$

All four statistics are feature-wise invariant for every j, so $d^t\big(\sigma(o_N^t)\big) = d^t\big(o_N^t\big)$. Since this holds for all permutations σ, d^t is permutation invariant in the agent ordering. □

A.2 Theorem 1: Well-Defined PLATO

A precondition to learning a rational policy in openness is producing *consistent actor queries* from finite parameters conditioned on a countably infinite observation space. In other words, the policy must be a well-defined function. GNN actors, such as MOHITO [Anil et al., 2025], meet this inherently because graph convolution is permutation invariant to node ordering and produces observation-specific action output from its hyperedges. Here we prove that PLATO also fulfills this requirement.

Theorem 1 (PLATO is well defined)**.** *Because all components of the actor and critic are permutation invariant, as shown in Lemma 1 and in the proof below, all countably infinite possible observations have only one categorical distribution in the actor and one value estimation in the critic.*

To prove PLATO is a well-defined function, we will show that the actor, π, and critic, V, have values for each state in this unbounded space,

$$(1)\forall s \in \mathbf{S}, \exists \Delta A \subseteq \mathbf{A} : (s, \Delta A) \in \pi \qquad (2)\forall s \in \mathbf{S}, \exists v \in \mathbb{R} : (s, v) \in V,$$

and that they consistently produce only one finite action distribution and v approximation for each state,

$$(3)\forall s \in \mathbf{S}, \quad \forall \Delta A_1, \Delta A_2 \subseteq \mathbf{A}\ s.t.\ \Delta A_1 \neq \Delta A_2 : (s, \Delta A_2) \notin \pi \text{ if } (s, \Delta A_1) \in \pi$$
$$(4)\forall s \in \mathbf{S}, \quad \forall v_1, v_2 \in \mathbb{R}\ s.t.\ v_1 \neq v_2 : (s, v_2) \notin V \text{ if } (s, v_1) \in V.$$

Proof (1). Assume there exists a state, $\hat{s} \in \mathbf{S}$ where $\pi(\hat{s}) \neq \Delta A \subseteq \mathbf{A}$. It must then be true that either $\pi(\hat{s})$ is undefined, in other words $\hat{s} \notin \pi$, or $\pi(\hat{s})$ must have support over an action not in $\mathbf{A}$.

PLATO's actor preprocessor factors agent observations into agent and task features of length $|X_i^t|$ and $|N^t|$ respectively. X_i^t refers to tasks agent i can act on; $X_i^t \subseteq X^t$. We assume this factorization is given by the environment and defined over Ψ_Ω. Regarding agent observations, team statistics produce $d^t \in \mathbb{R}^{4f_N}$, a *fixed*-length vector regardless of $|N^t|$. The subsequent MLP is defined over $\mathbb{R}^{4f_n}$. On the task observations, the task encoding applies individually to tasks and is thus defined over $\mathbb{R}^{f_x}$, and produces keys, $K^t \in \mathbb{R}^{(|X^t|+1)f_h}$. Each score $u_x^t = \mathbf{v}^\top \tanh(\mathbf{W}_K k_x^t + \mathbf{W}_q q^t)$ applies the same fixed-parameter function regardless of $|X_i^t|$ or $|N^t|$. Because the only input constraints come from Ψ_Ω, and Ψ_Ω is defined over all possible task and agent sets, $s \in \pi \iff s \in \mathbf{S}$. It must be the case that $\hat{s} \in \pi$.

There exists one score $u_x^t \forall X_i^t \in M^t$ and one NOOP score. PLATO assumes that there exists one action per task, $a_X \forall X_i$. There are also additional task-agnostic actions A_*, here a NOOP action. If the environment's action space is defined as $\Psi_A(...) = \prod_i \{a_x | x \in X_i, X_i \subseteq X\} + \{a_{NOOP}\}$, then any task score set must only include $a_x \forall X_i$ and NOOP by definition. It must be the case that $\pi(\hat{s}) \subseteq A$. Thus, by contradiction $\forall s \in \mathbf{S}, \exists \Delta A \subseteq \mathbf{A} : (s, \Delta A) \in \pi$. □

Proof (2). The global graph is a partition over the state, similar to the preprocessor's output, but encompassing all agents' local observations. If the factored agent, $\mathcal{N}$ and task nodes $\mathcal{X}$, have fixed feature dimensions, $f_\mathcal{N}, f_\mathcal{X}$, the critic's MLPs can encode them into a shared latent space, $\mathbb{R}^{f_h}$. The subsequent layers are GCNs over these nodes which are defined for a fixed f_h, global mean pooling which has no spatial requirements, and a decoder MLP, $MLP_v : \mathbb{R}^{f_h} \to \mathbb{R}$. Thus, if tasks and agents can be factored into state feature representations of fixed length regardless of change in the overall task and agent sets, then $\forall s \in \mathbf{S}, \exists v \in \mathbb{R} : (s, v) \in V$. □

Proof (3). Let $\sigma(o_i^t)$ represent a permutation of task and agent observations given by the environment. The preprocessor is not assumed to be permutation invariant, in that only the order of factors may change by the order of input observations. The agent encoder is permutation invariant because the team statistics are comprised of permutation-invariant operations, (mean, variance, minimum, maximum). The task encoder operates on all $o_x^t \in o_X^t$ independently and like the preprocessor preserves ordering, but K^t's values are unaffected by the order. The pointer decoder does not use a positional embedding, and more specifically,

$$\mathbf{v}^\top \tanh(\mathbf{W}_K \sigma(k_x^t) + \mathbf{W}_q q^t) = \mathbf{v}^\top \tanh(\mathbf{W}_K k_x^t + \mathbf{W}_q q^t) \quad \forall \sigma, x.$$

Therefore, π is permutation invariant, $\pi_\theta(\cdot \mid o_i^t) = \pi_\theta(\cdot \mid \sigma(o_i^t))$. Because there are no stochastic operations in π and we've established permutation invariance,

$$\forall s \in \mathbf{S}, \quad \forall \Delta A_1, \Delta A_2 \subseteq \mathbf{A} : (s, \Delta A_1), (s, \Delta A_2) \in \pi \implies \Delta A_1 = \Delta A_2.$$

□

Proof (4). Let $\sigma(\mathcal{G})$ represent a reordering of nodes describing the state given by the environment. Critic encoders, MLP_x and MLP_v, encode nodes independently and are thus permutation invariant to node ordering. GCN message passing only produces a different output graph if the input graph is different. By definition graph vertices are a set. Permuting their order does not change the graph. Because none of the remaining operations are stochastic and we have shown the critic is permutation invariant, $\forall s \in \mathbf{S}, \quad \forall v_1, v_2 \in \mathbb{R} : (s, v_1), (s, v_2) \in V \implies v_1 = v_2$. □

A.3 How the Pointer Policy Handles Task and Agent Openness

We characterize how the pointer actor handles each form of openness. We call the mechanism of Lemma 2, which points over tasks to select the action, the *task pointer*, and the mechanism of Lemma 3, which points over agents to build the query, the *agent pointer*. Lemma 2 concerns task openness (TO): the actor's action set is exactly the currently available tasks together with a NO-OP, and the relative preference between any two tasks is independent of which other tasks are present, so the policy adapts as tasks appear or disappear. Lemma 3 concerns agent openness (AO): the same pointing mechanism builds the query from the current agent set, with each observed agent encoded as a key and the deciding agent's own key acting as the query, so the policy adapts as agents arrive or depart.

Lemma 2 (Task openness: the task pointer is support-adaptive and rank-consistent)**.** *Fix parameters θ and let q be any fixed query vector, and assume PLATO's one-action-per-task setting: each task $x \in X^t$ has a single associated action a_x plus a persistent* NO-OP *(in the wildfire domain for example, $a_x = \textsf{suppress}(x)$). Let $\pi(\cdot) = \mathrm{softmax}\big([u_a^t]_{a \in X^t \cup \{\text{NO-OP}\}}\big)$ with per-task keys $k_x^t = \mathrm{MLP}_x(o_x^t)$ and scores $u_x^t = (\mathbf{v}^{\mathrm{t}})^\top \tanh(\mathbf{W}_K^{\mathrm{t}} k_x^t + \mathbf{W}_q^{\mathrm{t}} q)$, where $\mathbf{W}_K^{\mathrm{t}}$ and $\mathbf{W}_q^{\mathrm{t}}$ are learnable projection matrices for the keys and the query, respectively (note: the subscripts K and q name each matrix's role and are not indices; the upright superscripts* t *and* a *tag the task-side and agent-side scorers and are not the time index t), and $\mathbf{v}^{\mathrm{t}}$ is a learnable vector mapping the compatibility to a scalar score (cf. the Pointer Actor section of the main paper). Then (i) the support of π equals $\{\text{NO-OP}\} \cup \{a_x \mid x \in X^t\}$; and (ii) writing $\pi(x)$ for the mass on a_x, for any $x, x' \in X^t$ the log-odds $\log \frac{\pi(x)}{\pi(x')} = u_x^t - u_{x'}^t$ depend only on $(k_x^t, k_{x'}^t, q)$ and are invariant to $X^t \setminus \{x, x'\}$. Hence a task arriving or departing changes only the normalization and preserves the ranking of every remaining task.*

Proof. The task encoder maps each available task $x \in X^t$ independently to a key $k_x^t = \mathrm{MLP}_x(o_x^t)$, and a persistent NO-OP key is always present, so the score set is exactly $\{u_a^t : a \in X^t \cup \{\text{NO-OP}\}\}$ and the softmax over these $|X^t| + 1 \geq 1$ scores is a categorical whose support is $\{\text{NO-OP}\} \cup \{a_x \mid x \in X^t\}$, giving (i).

For (ii), the additive scorer [Bahdanau et al., 2014] $u_x^t = (\mathbf{v}^{\mathrm{t}})^\top \tanh(\mathbf{W}_K^{\mathrm{t}} k_x^t + \mathbf{W}_q^{\mathrm{t}} q)$ depends only on that task's key k_x^t and the fixed query q. With softmax normalizer $Z = \sum_{a \in X^t \cup \{\text{NO-OP}\}} e^{u_a^t}$ each task's probability is $\pi(x) = e^{u_x^t}/Z$ and $\pi(x') = e^{u_{x'}^t}/Z$. Forming the log-odds with the normalizer still in place and then cancelling the common Z,

$$\log \frac{\pi(x)}{\pi(x')} = \log \frac{e^{u_x^t}/Z}{e^{u_{x'}^t}/Z} = \log \frac{e^{u_x^t}}{e^{u_{x'}^t}} = u_x^t - u_{x'}^t.$$

Because each key $k_y^t = \mathrm{MLP}_x(o_y^t)$ is a function of task y's features alone and q is fixed, u_x^t and $u_{x'}^t$ do not depend on which other tasks are present; adding or removing a task $x'' \neq x, x'$ only inserts or deletes the term $e^{u_{x''}^t}$ in Z, leaving $u_x^t, u_{x'}^t$ and hence the log-odds unchanged. Hence $\log \frac{\pi(x)}{\pi(x')} = u_x^t - u_{x'}^t$ is invariant to $X^t \setminus \{x, x'\}$, establishing (ii); equivalently, each remaining task's score is unchanged, so their ranking is preserved and only the normalization shifts (content-based pointing [Vinyals et al., 2015]). The restriction $x'' \notin \{x, x'\}$ is the scope of this pairwise claim,

not a limitation: $\log \frac{\pi(x)}{\pi(x')}$ is defined only while both x and x' lie in the support of (i). Removing either leaves the policy well defined and rank-consistent among the tasks that remain, and since each key $k_x^t = \mathrm{MLP}_x(o_x^t)$ is recomputed from current features, a task that departs and later reappears re-enters under the same log-odds $u_x^t - u_{x'}^t$, carrying no state during its absence. □

Lemma 3 (Agent openness: the agent pointer is support-adaptive and rank-consistent)**.** *Fix parameters θ, let i be the deciding agent, and let the* team *N^t be the set of agents currently observed by i: each element $j \in N^t$ is one currently present agent, with observed features o_j^t accounting for one row of the agent-feature matrix o_N^t. Both are part of agent i's local observation o_i^t, so a different deciding agent generally has a different agent-feature matrix; since the whole lemma is computed by the fixed deciding agent i, we omit the extra subscript i on o_j^t, o_N^t, and all derived quantities (k_j^t, e_j^t, α^t, q^t) for readability. An agent always observes itself, so $i \in N^t$ and $|N^t| \geq 1$. Let per-agent keys $k_j^t = \mathrm{MLP}_n(o_j^t) \in \mathbb{R}^{f_h}$ for $j \in N^t$, scores $e_j^t = (\mathbf{v}^{\mathrm{a}})^\top \tanh(\mathbf{W}_K^{\mathrm{a}} k_j^t + \mathbf{W}_q^{\mathrm{a}} k_i^t)$ with the deciding agent's own key k_i^t as the query, where $(\mathbf{W}_K^{\mathrm{a}}, \mathbf{W}_q^{\mathrm{a}}, \mathbf{v}^{\mathrm{a}})$ are the agent-side counterparts of the task-side $(\mathbf{W}_K^{\mathrm{t}}, \mathbf{W}_q^{\mathrm{t}}, \mathbf{v}^{\mathrm{t}})$ of Lemma 2: learnable projection matrices for the keys and the query, and a learnable scoring vector (cf. the Pointer Actor section of the main paper), attention $\alpha^t = \mathrm{softmax}\big([e_j^t]_{j \in N^t}\big)$, and the query, the attention-weighted read of the agent keys, $q^t = \sum_{j \in N^t} \alpha_j^t k_j^t$. Then (i) α^t is a well-defined categorical whose support is exactly N^t, for every $|N^t| \geq 1$; (ii) $q^t \in \mathbb{R}^{f_h}$ is a fixed-length, permutation-invariant function of the agent set, for every $|N^t| \geq 1$; and (iii) for any $j, j' \in N^t$ the log-odds $\log \frac{\alpha_j^t}{\alpha_{j'}^t} = e_j^t - e_{j'}^t$ depend only on $(k_j^t, k_{j'}^t, k_i^t)$ and are invariant to $N^t \setminus \{j, j', i\}$. Hence an agent arriving or departing changes only the normalization and preserves the* relative weighting *of every remaining agent, that is, the ratio $\alpha_j^t / \alpha_{j'}^t$ between the attention weights of any two agents that remain. Moreover, the agent set enters each task score $u_x^t = (\mathbf{v}^{\mathrm{t}})^\top \tanh(\mathbf{W}_K^{\mathrm{t}} k_x^t + \mathbf{W}_q^{\mathrm{t}} q^t)$, with $k_x^t = \mathrm{MLP}_x(o_x^t)$ the key of task x and $(\mathbf{W}_K^{\mathrm{t}}, \mathbf{W}_q^{\mathrm{t}}, \mathbf{v}^{\mathrm{t}})$ the task-side learnable projections and scoring vector of Lemma 2, where the subscript x indexes a task just as j indexes an agent (u_x^t is the task-side score, e_j^t its agent-side counterpart), only through q^t, and because Lemma 2 holds for any fixed query, its conclusions apply verbatim with $q = q^t$ at each t.*

Proof. For (i), the agent encoder maps each observed agent $j \in N^t$ independently to a key $k_j^t = \mathrm{MLP}_n(o_j^t)$, and the deciding agent always observes itself, so $i \in N^t$ and the score set $\{e_j^t : j \in N^t\}$ contains at least the ego score; the softmax over these $|N^t| \geq 1$ scores is a categorical whose support is exactly N^t.

For (ii), permutation invariance of the normalizer $Z_N = \sum_{l \in N^t} e^{e_l^t}$ and the read $q^t = \sum_{j \in N^t} \alpha_j^t k_j^t$ is exactly Lemma 1. Since q^t is a convex combination of keys in $\mathbb{R}^{f_h}$, its length is f_h regardless of $|N^t|$, giving (ii).

For (iii), the additive scorer [Bahdanau et al., 2014] $e_j^t = (\mathbf{v}^{\mathrm{a}})^\top \tanh(\mathbf{W}_K^{\mathrm{a}} k_j^t + \mathbf{W}_q^{\mathrm{a}} k_i^t)$ depends only on that agent's key k_j^t and the deciding agent's key k_i^t, which is always available since $i \in N^t$. With softmax normalizer $Z_N = \sum_{l \in N^t} e^{e_l^t}$ each agent's weight is $\alpha_j^t = e^{e_j^t}/Z_N$ and $\alpha_{j'}^t = e^{e_{j'}^t}/Z_N$. Forming the log-odds with the normalizer still in place and then cancelling the common Z_N,

$$\log \frac{\alpha_j^t}{\alpha_{j'}^t} = \log \frac{e^{e_j^t}/Z_N}{e^{e_{j'}^t}/Z_N} = \log \frac{e^{e_j^t}}{e^{e_{j'}^t}} = e_j^t - e_{j'}^t.$$

Because each key $k_l^t = \mathrm{MLP}_n(o_l^t)$ is a function of agent l's features alone, e_j^t and $e_{j'}^t$ do not depend on which other agents are present; adding or removing an agent $j'' \notin \{j, j', i\}$ only inserts or deletes the term $e^{e_{j''}^t}$ in Z_N, leaving $e_j^t, e_{j'}^t$ and hence the log-odds unchanged. Hence $\log \frac{\alpha_j^t}{\alpha_{j'}^t} = e_j^t - e_{j'}^t$ is invariant to $N^t \setminus \{j, j', i\}$, establishing (iii); equivalently, each remaining agent's score is unchanged, so their relative weighting is preserved and only the normalization shifts (content-based pointing [Vinyals et al., 2015]). Here *unchanged* means at the same timestep with the same observed features: the scorer reads only (k_j^t, k_i^t), never the rest of N^t, so a change in the team membership by itself cannot re-rank the remaining agents. The policy still adapts at once, because the departed key drops out of q^t and the weights renormalize, shifting the task distribution immediately; at later steps the

departure also changes what agents observe (in wildfire, an unattended fire grows and suppressant use shifts), and those new features change the scores. The restriction $j'' \notin \{j, j', i\}$ is the scope of this pairwise claim, not a limitation: i never departs and its key is part of the stated dependence, while $\log \frac{\alpha_j^t}{\alpha_{j'}^t}$ is defined only while both j and j' lie in the support of (i). Removing either leaves the attention well defined and rank-consistent among the agents that remain, and since each key $k_j^t = \mathrm{MLP}_n(o_j^t)$ is recomputed from current features, an agent that departs (for example, after running out of suppressant) and later returns is re-encoded from its current features and re-enters under the same scoring rule: its log-odds against any current teammate are determined by its current features and the current ego key k_i^t. The policy carries no memory of the agent through its absence: nothing is saved at departure and nothing is restored at re-entry; any persistence (for example, its suppressant level) lives in the environment state and reaches the policy only through the agent's current observed features.

For the final claim, each task key $k_x^t = \mathrm{MLP}_x(o_x^t)$ is a function of task features alone, so the agent set enters the score $u_x^t = (\mathbf{v}^{\mathrm{t}})^\top \tanh(\mathbf{W}_K^{\mathrm{t}} k_x^t + \mathbf{W}_q^{\mathrm{t}} q^t)$ only through q^t. At any time t, the read q^t is one fixed vector in $\mathbb{R}^{f_h}$. Lemma 2 was proved for an arbitrary fixed query, so its conclusions hold verbatim with $q = q^t$. In particular, q^t is computed from the agent features o_N^t alone, so it does not change when tasks arrive or depart, and Lemma 2's invariance to the rest of the task set applies. Note that q^t itself does change when the team changes: the weights renormalize and keys enter or leave the sum. That is the intended adaptation to the current team; claim (iii) asserts invariance of the relative weighting among the remaining agents, not constancy of q^t.

□

A.4 Implications of the Openness Lemmas

Together, Lemmas 2 and 3 pin down how the policy reacts when the task set or the team changes: adaptation flows through the observed features and the query, never through the membership of the sets themselves. We illustrate with an example. Let the team be $N^t = \{n_1, n_2, n_3\}$, three agents as defined in Lemma 3, and let the task set X^t contain the fires x_1, x_2, and x_5, where fire x_1 requires all three agents while the smaller fires x_2 and x_5 can each be handled by one agent. Suppose n_2 departs, for example after running out of suppressant. At that instant, by Lemma 3(iii), the relative weighting between n_1 and n_3 is unchanged: the departure by itself does not re-rank the remaining agents. The policy nevertheless adapts immediately: n_2's key $k_{n_2}^t$ drops out of the query q^t and the attention weights α^t renormalize, so each remaining agent's task distribution shifts, and by Lemma 2 this shift re-weights the available tasks without disturbing the support or requiring retraining; n_1 may now prefer x_2 and n_3 may prefer x_5, so the joint attack on x_1 dissolves. In the following steps the consequences of the departure enter through the features: x_1 grows unattended and suppressant use shifts, and these new observations change both the agent scores and the task scores. If n_2 later returns, or a brand-new agent arrives, it is encoded from its current features and joins the same computation, with no reserved slot, no reset, and no retraining. The two lemmas thus guarantee that openness changes the policy only in the intended way: the distribution is always well defined over exactly the current tasks and the current team, and it reacts to openness through content rather than through identity or count.

A.5 Theorem 2: Computational complexity

Here we analyze the space and time complexity of PLATO. Our time complexity bound is non-deterministic, in that we consider parallelization where possible. We treat all fixed hyperparameters (f_N, f_X, f_h, number of GCN layers), including the feature size of factored tasks and agents as constants. We do this to express complexity solely in terms of the quantities that vary under openness, the number of active agents $|N^t|$ and active tasks $|X^t|$.

Theorem 2 (Complexity Analysis for PLATO)**.** *Assuming neural network hyperparameters, feature dimensions, and graph convolutional layers are constants, the per-timestep complexity of PLATO is* $\mathcal{O}(|N^t|^2 + |N^t||X^t|)$.

Actor. The preprocessor and team statistics scan o_N^t and o_X^t in $O(|N^t|+|X^t|)$. The team-statistics step outputs a *fixed*-length descriptor $d^t \in \mathbb{R}^{4f_N}$ regardless of $|N^t|$, so MLP_s contributes an $O(1)$

constant. MLP_x is applied independently to each of the $|X^t| + 1$ task features, costing $O(|X^t|)$; because tasks are encoded without shared state, this step parallelizes fully over the task dimension. One-step additive attention [Bahdanau et al., 2014, Vinyals et al., 2015] computes a scalar score per key in $O(1)$ and runs over all keys in $O(|X^t|)$; SOFTMAX normalization is likewise $O(|X^t|)$. The per-agent actor cost is $O(|N^t| + |X^t|)$, and for all $|N^t|$ agents sharing the same policy the per-timestep actor cost is $O(|N^t|(|N^t| + |X^t|))$.

Critic. Node encoding via separate MLPs costs $O(|N^t| + |X^t|)$. The bipartite graph $\mathcal{G}^t$ has $|\mathcal{E}^t| = |N^t||X^t|$ edges; each GCN layer [Kipf and Welling, 2016] aggregates over $|\mathcal{E}^t|$ edges in $O(|N^t||X^t|)$, which dominates the per-node projection $O(|N^t| + |X^t|)$. With $L = 2$ layers and a residual connection [He et al., 2016], total propagation is $O(|N^t||X^t|)$. Mean pooling and the value head are $O(|N^t| + |X^t|)$ and $O(1)$, respectively. The critic cost per timestep is therefore $O(|N^t||X^t|)$, which dominates the actor during training. At execution time the critic is absent, and the cost across all agents reduces to $O(|N^t|(|N^t| + |X^t|))$ per timestep.

A.6 Reward bounds in Wildfire

In Wildfire, the only positive rewards are r^{size}_{putout} for successfully suppressing a fire, and the only negative rewards are $r^{\text{size}}_{burnout}$ when a fire burns out. With stochastic random ignition, in the worst case all fires immediately ignite once they burn out. Therefore, trivially, the worst penalty over any trajectory τ is bounded by the rate fires burn out,

$$R(\tau) \geq \left\lfloor \frac{h}{\text{int}_0 + 1} \right\rfloor \sum_x r^{size_x}_{burnout}, \tag{1}$$

where h is the time horizon, and int_0 is the initial intensity of fires. The $+1$ is required because fires only ignite when they have int $= 0$, so there is a timestep delay.

For the upper bound, fire sizes are very important. All fires start at the same intensity, int_0, so agents, N, strictly benefit from collaborating if $r^{size}_{putout} \geq \text{size} \cdot r_{putout}$. When fires are all size $= 1$, the maximum reward can similarly be bounded by the maximum rate fires can be put out,

$$R(\tau) \leq \frac{h}{\text{int}_0} \cdot r^{\text{size}_x}_{putout} \cdot \min(|N|, \lceil \text{number of unique reachable fires}/2 \rceil). \tag{2}$$

In this bound, the set of *unique reachable fires* is all fires reachable by at least one unique agent. This approach will overestimate the impact of interleaving fires. Size 1 fires can be suppressed by one agent, so the optimal strategy is to repeatedly put out either the one fire available to that agent ($|N|$), or if agents can reach two unique fires, alternate fighting them to avoid the one timestep ignition delay, ($\lceil \text{number of unique reachable fires}/2 \rceil$). When grid size is sufficiently large, increasing it will not change *unique reachable fires* if agent position and reach are fixed.

We address both fire size, and *fuel*, a mechanic added to some recent uses of the Wildfire environment, [Patino et al., 2025], through optimization. Fuel involves putting out or letting a fire burn out deterministically decreases the number of times it can ignite. Our algorithm solves this by calculating optimal cycles. Let agents repeatedly put out fires in *cycles*, $c \in C$. Agents may pause one cycle while waiting for reignition to work on another. Each cycle contains one fire, with fire size monotonically increasing by cycle index.

We determine which cells will be completed as an integer program solution, similar to the Maximum Temporal Matching Problem [Mertzios et al., 2023],

$$
\begin{aligned}
\text{Maximize} \quad & \sum_{c \in C} \sum_{x \in X} f_{x,c} \cdot r_x \\
\text{Such That} \quad & \sum_{n \in N} a_{n,x,c} = f_{x,c} \cdot \text{size}_x && \forall x \in X, c \in C && (3) \\
& \sum_{x \in X} a_{n,x,c} \cdot \text{size}_x \geq \sum_{x \in X} a_{n,x,c+1} \cdot \text{size}_x && \forall n \in N, c < |C| && (4) \\
& \sum_{c \in C} a_{n,x,c} \leq 1 && \forall n \in N, x \in X && (5) \\
& \sum_{x \in X} a_{n,x,c} \leq 1 && \forall n \in N, c \in C && (6) \\
& f_{x,c}, a_{n,x,c} \in \{0,1\} && \forall n, x, c.
\end{aligned}
$$

The constraints on this integer program (IP) solution ensure that (3) fires have sufficient collaboration, (4) monotonic cycle fire size increase, (5) agents attack unique fires across cycles, and (6) agents attack only one fire per cycle. The number of cycles is bounded, $|C| \leq$ num reachable fires. Now we determine how many times each cycle occurs. We handle this post to avoid solving the integer program over the whole horizon. Let ρ_c be the number of times cycle c produces a reward. If,

$$
\begin{aligned}
\rho_{x,1} &= \min(\text{fuel}_x, \left\lfloor \frac{h}{\text{int}_0} \right\rfloor) \\
\rho_{x,c} &= \min(\text{fuel}_x, \left\lfloor \frac{h}{\text{int}_0} \right\rfloor - \sum_{c'=1}^{c-1} \rho_{x,c'}),
\end{aligned}
\tag{7}
$$

then $R(\tau) \leq \sum_{n,x,c} f_{x,c} \cdot r_x \cdot \rho_{x,c}$, for any stochastic Wildfire environment.

B Reproducibility

Here we formally define the Wildfire environment. We also include all hyperparameters, our seeding approach, and hyperparameter tuning methodology. Our implementation is publicly available at https://github.com/oasys-mas/PLATO.

Table 5 lists the best tuned hyperparameters used for each learned method.

Parameter	PLATO (MLP)	PLATO (LSTM)	MOHITO	DGN	DICG	DeepSets
Network Architecture						
Hidden Dimension (Actor)	128	128	32	128	128	128
Hidden Dimension (Critic)	128	128	32	128	128	128
Activation Function	ReLU, Leaky ReLU	ReLU, Leaky ReLU	ReLU	ReLU	Tanh	-
Learning Rates & Optimization						
Actor Learning Rate	5e-4	5e-4	1e-4	5e-3	1e-4	5e-4
Critic Learning Rate	9e-4	9e-4	5e-4	1e-4	1e-4	9e-4
Optimizer	Adam	Adam	Adam	RMSprop	Adam	Adam
Optimizer Epsilon	1e-5	1e-5	1e-8	1e-5	1e-5	1e-5
Gradient Clipping	1.0	1.0	1.0	10.0	10.0	1.0
Input Processing						
Observation Padding	No	No	No	Yes	Yes	No
Padding Value	-	-	-	-1.0	-1	-
Data Collection						
Batch Size	2048	2048	16	1024	1024	2048
Parallel Environments	1	1	1	1	1	1

Table 5: Configuration: Hyperparameters for all methods. This shows the best hyperparameters from tuning.

Compute resources. All experiments were conducted on a high-performance computing cluster. The experiments were executed on Linux-based compute nodes running a 64-bit x86 architecture

with dual Intel Xeon Gold 6348 CPUs at 2.60 GHz, for a total of 56 CPU cores. The cluster nodes used the Linux kernel version 4.18.0-553.89.1.el8_10.x86_64. Experiments were performed strictly on CPU compute using the free-range-zoo library introduced by Patino et al. [2025]. Each individual training run (one method, one grid, one setup) takes approximately 3–5 days of CPU time. The full experimental suite (covering all methods, grids, setups, and Optuna tuning trials) required approximately 6,000–10,000 CPU-days in total.

Hyperparameter tuning. We tune all methods with Optuna [Akiba et al., 2019] using a Tree-structured Parzen Estimator (TPE) sampler with a shared budget of 50 trials per method. Each trial trains for 2,000 episodes and is evaluated on 15 validation episodes; the trial objective is the mean validation return on the 3×3 grid under S3, the most challenging setup. For each method, the search space is anchored to the default hyperparameters reported in its published codebase, with ranges that bracket those defaults on a log scale for learning rates and a linear scale for bounded scalars. Specifically, learning rates are searched over $[10^{-5}, 10^{-2}]$ (PLATO, DICG) or $[10^{-5}, 10^{-3}]$ (MOHITO) and $[10^{-4}, 10^{-2}]$ (DGN); hidden dimensions over $\{64, 128, 256\}$ (PLATO, DGN, DICG) or $\{32, 64, 128\}$ (MOHITO); batch sizes over $\{512, 1024, 2048, 4096\}$ (PLATO), $\{256, 512, 1024, 2048\}$ (DGN), $\{512, 1024, 2048\}$ (DICG), or $\{16, 32, 64, 128\}$ (MOHITO); and entropy coefficients, gradient clipping, and exploration schedules similarly bracketing each method's reported defaults. This ensures each baseline is given a fair and equal opportunity to find its best configuration in our environment, independently of its original domain.

Seeding protocol. We use a three-tier seeding protocol, following established reproducibility practice in deep RL [Henderson et al., 2018] with respect to openness. The **global seed** (42) initializes all network parameters and algorithm-level randomness (PyTorch, NumPy, Python random) once per run. The **train seed** (300) resets the environment at the start of every training episode, so the policy is always trained from a consistent initial fire configuration; within-episode stochasticity (fire spread, random ignitions, and stochastic refills) still varies across episodes, exposing the policy to diverse dynamics under a fixed start. The **execution seed** jointly seeds both the environment and all random-number generators at the start of each evaluation episode, making every episode exactly reproducible from its seed alone; we assign $N{=}50$ distinct sequential seeds per checkpoint $(200, 201, \ldots, 249)$. All three seed ranges are disjoint by construction, so the policy is never evaluated on conditions seen during training. Sharing the same execution seeds across all methods yields seed-matched episode-level observations, which controls for seed-induced variance and increases the sensitivity of the Shapiro-Wilk-adaptive two-sided tests [Wilcoxon, 1945, Shapiro and Wilk, 1965] used for significance.

C Additional Results

C.1 Wildfire Suppression

We simulate *task openness* and *agent openness* using the domain of wildfire suppression. The WILDFIRE domain [Patino et al., 2025] was developed and validated within OASYS, provides a publicly available codebase with documented configurations, and natively exhibits both AO (agents exit and re-enter as suppressant depletes) and both forms of TO (endogenous spread and exogenous random ignitions) without artificial modification, while supporting varied grid sizes for systematic generalization analysis.

Each firefighter, N_i, has a suppression range, a finite suppressant level, and equipment health. Agents cannot move between cells; they act only on fires within range. When agent i depletes its suppressant, it exits to recharge suppressant or repair its equipment and may later return, introducing *agent openness*.

C.2 Setup Figures

We define *native* (i.e., training and testing on the same grid; see Figs. 3–5) and *zero-shot* (i.e., training and testing on different grids; see Fig. 6). Zero-shot evaluates generalization to unseen environments, where agents must handle unseen configurations, new spatial layouts, and different coordination patterns without retraining. The same team from training is used, but on a different grid with a different number of initial and possible fires and novel positions: 2×3 (3 agents, 2 fires)

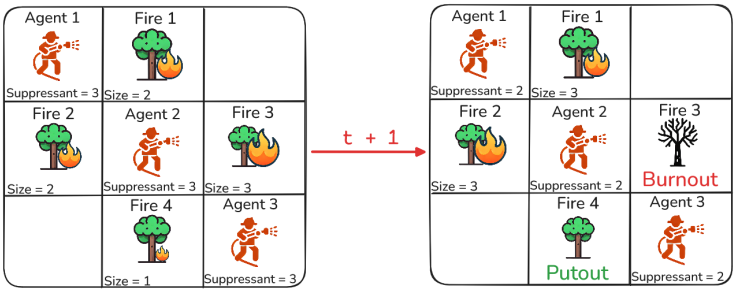

Figure 2: WILDFIRE example (3×3). Fires (tasks X_j) ignite, spread, burn out, or are put out (i.e., extinguished) while agents (N_i) deplete suppressant or leave for repairs. Both sets evolve over time, creating an open multi-agent system.

$\rightarrow 3 \times 3$ (3 agents, 5 fires) and 4×4 (3 agents, 7 fires); 3×3 (3 agents, 4 fires) $\rightarrow 4 \times 4$ (3 agents, 7 fires) and 5×5 (3 agents, 8 fires).

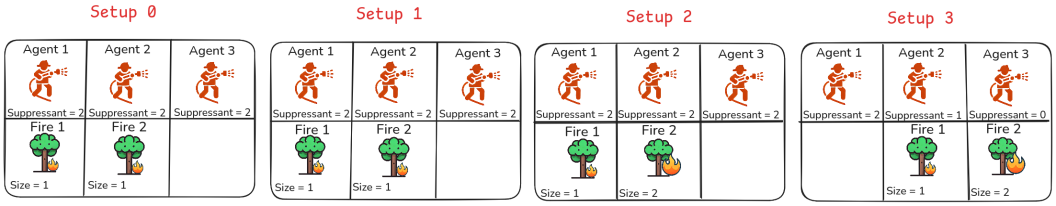

Figure 3: Setups (S0–S3) on 2×3 showing the initial suppressant level that each firefighter agent has and the fire size. Setup 1 enables fire spread (endogenous TO). Setup 2 replaces spread with random ignitions (exogenous TO) and introduces a larger fire type, testing a qualitatively different openness mechanism. Setup 3 is the most demanding: it reintroduces fire spread alongside random ignitions and adds uneven initial suppressant levels, combining both TO mechanisms with AO resource constraints. Setup 0 uses the same initial conditions as Setup 1 but disables both task openness and agent openness, yielding a static, non-open setting.

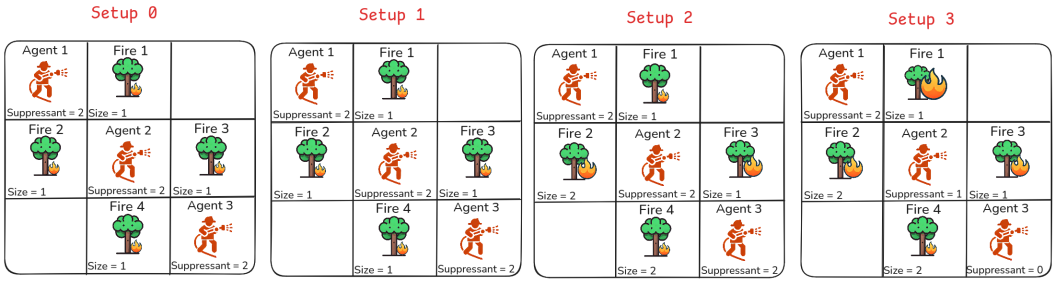

Figure 4: Setups (S0–S3) on 3×3 showing the suppressant level that each firefighter agent has and the fire size. Setup 1 enables fire spread (endogenous TO). Setup 2 replaces spread with random ignitions (exogenous TO) and introduces two larger fire types, testing a distinct openness mechanism. Setup 3 reintroduces fire spread alongside random ignitions and adds uneven initial suppressant levels, combining both TO mechanisms with AO resource constraints.

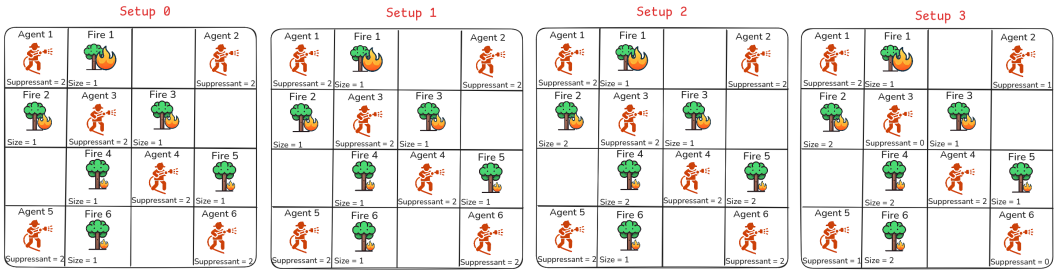

Figure 5: Setups (S0–S3) on 4×4 showing the suppressant level that each firefighter agent has and the fire size. Setup 1 enables fire spread (endogenous TO). Setup 2 replaces spread with random ignitions (exogenous TO) and introduces two larger fire types, testing a distinct openness mechanism. Setup 3 reintroduces fire spread alongside random ignitions and adds uneven initial suppressant across all six agents, combining both TO mechanisms with more heterogeneous AO.

C.2.1 Zero-Shot Evaluation

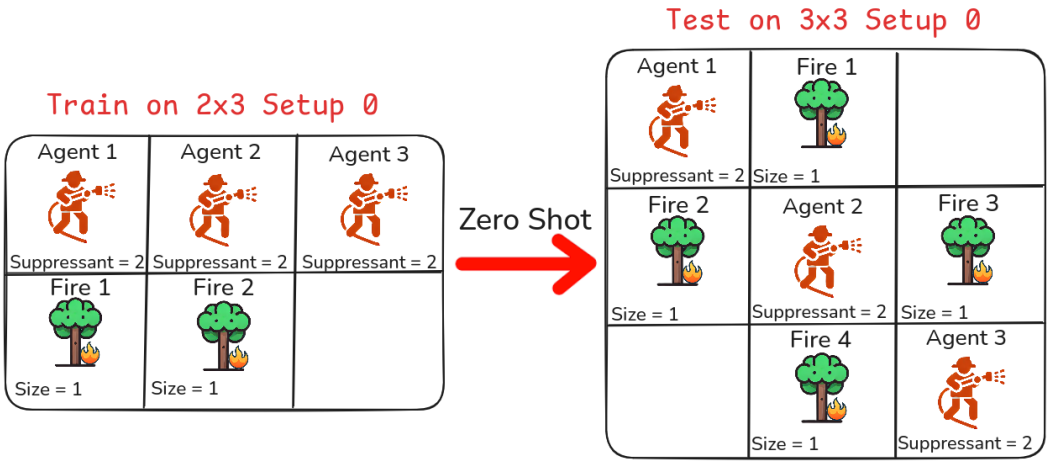

Figure 6: Example zero-shot evaluation. Agents are trained on a 2×3 Setup 0 environment and tested without retraining on a larger 3×3 Setup 0 environment.

C.3 Domain and Openness Setups

Setup	Fire spread	Random ignitions	Refill success
S0	× (0.0)	× (0.0)	-
S1	✓ (0.6)	× (0.0)	1.0
S2	× (0.0)	✓ (0.6)	0.8
S3	✓ (0.6)	✓ (0.6)	0.6

Table 6: Configuration: Openness mechanisms across setups. Endogenous TO is fire spread, exogenous TO is random ignitions, and AO is stochastic refill success. Numbers indicate the probability per step. In S0 agents have unlimited suppressant (suppressant decrease disabled).

C.4 Setup Details and Openness Statistics

Table 7 summarizes how frequently openness events occur within the 100-step horizon, and Table 8 summarizes the grid-scale configuration differences.

Grid	Setup			
	S0	S1	S2	S3
Zero-suppressant count (agents ran out of suppressant)				
2×3	0.00±0.00	16.25±12.35	56.05±28.56	54.06±34.85
3×3	0.00±0.00	10.69±8.07	64.71±27.87	61.78±42.16
4×4	0.00±0.00	14.91±10.98	60.52±55.31	54.40±60.45
New fires count				
2×3	0.00±0.00	3.16±6.46	30.46±15.74	19.53±15.81
3×3	0.00±0.00	2.17±2.92	34.27±16.93	22.85±18.64
4×4	0.00±0.00	1.97±2.92	33.60±34.40	21.97±28.37

Table 7: Openness Event Statistics: Execution openness event frequency within the fixed 100-step episode horizon, aggregated across all models (mean ± std). S0 is closed, hence no ignitions and no zero-suppressant events. In S1–S3, both task openness (new fires) and agent openness (zero-suppressant leading to exit/refill) occur frequently.

	2×3	3×3	4×4
Cells (rc)	6	9	16
Agents ($\lvert N_0 \rvert$)	3	3	6
Initial fires ($\lvert X_0 \rvert$)	2	4	6
Init suppressant (S0–S2)	[2,2,2]	[2,2,2]	[2,2,2,2,2,2]
Init suppressant (S3)	[2,1,0]	[2,1,0]	[2,1,0,2,1,0]
Fire types (S0–S1)	1×2	1×4	1×6
Fire types (S2)	1×1, 2×1	1×2, 2×2	1×4, 2×2
Fire types (S3)	1×1, 2×1	1×2, 2×2	1×3, 2×3

Table 8: Configuration: Grid details across sizes.

C.5 Configuration Tables

Table 9 enumerates the full grid-specific instantiations for each setup.

Setup	**2×3**	**3×3**	**4×4**
0	Agents: 3 Initial suppressant: [2,2,2] Initial fires: 2 Fire types: 1×2	Agents: 3 Initial suppressant: [2,2,2] Initial fires: 4 Fire types: 1×4	Agents: 6 Initial suppressant: [2,2,2,2,2,2] Initial fires: 6 Fire types: 1×6
1	Agents: 3 Initial suppressant: [2,2,2] Initial fires: 2 Fire types: 1×2	Agents: 3 Initial suppressant: [2,2,2] Initial fires: 4 Fire types: 1×4	Agents: 6 Initial suppressant: [2,2,2,2,2,2] Initial fires: 6 Fire types: 1×6
2	Agents: 3 Initial suppressant: [2,2,2] Initial fires: 2 Fire types: 1×1, 2×1	Agents: 3 Initial suppressant: [2,2,2] Initial fires: 4 Fire types: 1×2, 2×2	Agents: 6 Initial suppressant: [2,2,2,2,2,2] Initial fires: 6 Fire types: 1×4, 2×2
3	Agents: 3 Initial suppressant: [2,1,0] Initial fires: 2 Fire types: 1×1, 2×1	Agents: 3 Initial suppressant: [2,1,0] Initial fires: 4 Fire types: 1×2, 2×2	Agents: 6 Initial suppressant: [2,1,0,2,1,0] Initial fires: 6 Fire types: 1×3, 2×3

Table 9: Configuration: Grid-specific instantiations for each setup, including the number of agents, initial suppressant levels, initial fires, and initial fire types.

C.6 Convergence, checkpoint selection, and statistical testing

In closed settings, a common stability criterion is

$$\left|\bar{R}_t - \bar{R}_{t-W}\right| < \varepsilon, \tag{8}$$

where $\bar{R}_t$ is the moving-average return at episode t; W is a smoothing window; ε is a small stability threshold.

In open settings, episode returns vary under different realizations of the task and agent sets, so convergence is assessed over an evaluation distribution. For a saved checkpoint θ_k, we estimate its expected performance as

$$\hat{\mu}_k = \frac{1}{|\mathcal{S}|N} \sum_{s \in \mathcal{S}} \sum_{i=1}^{N} R(\theta_k; s, i), \tag{9}$$

where $\mathcal{S}$ is the set of setups; N is the number of evaluation seeds per setup; $R(\theta_k; s, i)$ is the episode return of checkpoint θ_k on setup s and seed i.

We select the top three checkpoints for each method by $\hat{\mu}_k$ over the second half of training and report mean $\pm$ std over their aggregated evaluation episodes.

Statistical test procedure. We use a **Shapiro-Wilk-adaptive two-sided test** [Wilcoxon, 1945, Shapiro and Wilk, 1965] for all significance claims: for each comparison, we apply a Shapiro-Wilk normality pre-test (at the same Bonferroni-corrected α as the primary test) to the per-seed differences; if the test fails to reject normality ($p \geq \alpha$), we use the two-sided paired t-test; if it rejects normality ($p < \alpha$), we use the two-sided Wilcoxon signed-rank test. Because every method is evaluated on the same fixed seed set, each seed produces one matched observation per method under identical stochastic conditions; pairing by seed removes this shared variance and is strictly more powerful than an unpaired test on the same data [Montgomery, 2017]. For the empirically best model a and any competitor b, we form per-seed differences over their shared seed set $\mathcal{S}_{ab}$,

$$t = \frac{\bar{\delta}}{s_\delta / \sqrt{|\mathcal{S}_{ab}|}}, \qquad \bar{\delta} = \frac{1}{|\mathcal{S}_{ab}|} \sum_{s \in \mathcal{S}_{ab}} \delta_s, \quad \delta_s = r_a(s) - r_b(s), \tag{10}$$

where s_δ is the sample standard deviation of $\{\delta_s\}$; differences are negated for cost metrics (lower is better). A model is **bolded** only when it significantly outperforms *every* other model in the comparison family; no cell is bolded when no model qualifies.

Bonferroni correction and comparison families. We apply Bonferroni correction $\alpha = 0.05/k$ [Hochberg and Tamhane, 1987], where k is the number of models in the *comparison family* for each table. Bonferroni requires no assumptions on the dependence structure among tests and is the most conservative choice; with the small k values used here, the power cost is negligible. We define three families, each scoped to the scientific question of the corresponding table or figure (Table 10):

- **Main results: all native grids** (Tables 1, 2, and appendix metric tables): all $k = 9$ models (PLATO (MLP-Additive), DGN, MOHITO, DICG, DeepSets, NOOP, Random, Weakest, Strongest) giving $\alpha = 0.05/9 = 0.00556$. The family is per table (per metric, per grid), consistent with standard practice in empirical MARL [Demšar, 2006].
- **Zero-shot generalization** (Table 3, Appendix C.7.2): naive baselines have no learned policy and do not participate in the generalization question; the family is $k = 5$ smart models (PLATO, DGN, MOHITO, DICG, DeepSets), giving $\alpha = 0.05/5 = 0.01$.
- **Ablation study** (Table 4 in main text; additional metrics in Appendix C.8): only PLATO variants are compared; the $k = 4$ variants are (mlp+add) (as reference), (mlp+dot), (lstm+add), and (lstm+dot), giving $\alpha = 0.05/4 = 0.0125$.

C.7 Detailed Results

This appendix provides detailed results on performance and efficiency (burnouts, putouts, NOOP%, and reward per fight) for all experimental configurations. These metrics complement the episode

Family	**Models included**	k	$\alpha = 0.05/k$	p-threshold
Native results	PLATO, DGN, DICG, MOHITO, DeepSets, NOOP, Random, Weakest, Strongest	9	$0.05/9$	0.00556
Zero-shot	PLATO, DGN, DICG, MOHITO, DeepSets (smart models only; naive baselines excluded)	5	$0.05/5$	0.01000
Ablation	4 PLATO variants: mlp+add, mlp+dot, lstm+add, lstm+dot	4	$0.05/4$	0.0125

Table 10: Bonferroni correction families. For each family, $\alpha = 0.05/k$ is the per-test significance threshold applied per setup per table. A bold entry means the empirically best model significantly outperforms *all* others in the family at that threshold.

	2×3			
Model	S0	S1	S2	S3
PLATO	4.00±0.00	**52.84±25.62**	**68.17±17.80**	76.07±10.73
DICG	4.00±0.00	18.92±14.81	51.76±12.71	70.80±7.81
DGN	3.92±0.56	14.41±11.03	45.77±15.29	69.05±10.93
MOHITO	4.00±0.00	9.67±7.30	8.08±14.66	58.20±14.46
DeepSets	4.00±0.00	9.96±6.25	58.51±25.03	**79.25±7.34**
NOOP	−4.00±0.00	−4.00±0.00	−6.00±0.00	−6.00±0.00
Random	3.92±0.57	4.36±2.27	3.56±11.24	34.48±18.00
Weakest	4.00±0.00	3.68±0.74	64.00±0.00	64.00±0.00
Strongest	4.00±0.00	5.24±3.68	24.44±25.75	63.12±1.08

Table 11: Performance: Episode return (mean ± std), trained and tested on 2×3. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

	2×3			
Model	S0	S1	S2	S3
PLATO	1.27±0.45	0.52±0.12	0.67±0.17	0.58±0.10
DICG	1.12±0.37	0.48±0.31	0.50±0.14	0.56±0.07
DGN	1.15±0.54	0.41±0.41	0.46±0.19	0.55±0.10
MOHITO	1.04±0.22	0.58±0.50	−0.07±0.65	0.49±0.11
DeepSets	1.21±0.55	0.74±0.49	0.71±0.18	0.59±0.06
NOOP	–	–	–	–
Random	1.15±0.54	0.79±0.68	−0.08±0.33	0.38±0.13
Weakest	1.21±0.28	1.15±0.43	0.53±0.00	0.43±0.00
Strongest	1.21±0.28	**1.19±0.34**	0.10±0.30	0.43±0.01

Table 12: Efficiency: Reward per fight (mean ± std), trained and tested on 2×3. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$). NOOP takes no fight actions, so its reward per fight is undefined (–).

returns reported in the main text and offer insight into the behavioral differences between methods. Putouts measure successful task completion. Burnouts measure failures (fires reaching terminal intensity). NOOP% measures action efficiency and implicit idleness induced by AO or conservative policies.

C.7.1 Native Training: All Metrics

PLATO refers to the mlp+add variant unless otherwise noted. Tables 11 and 12 report episode return and reward per fight on 2×3 (referred to from the main text). Tables 13–15 report NOOP, burnouts, and putouts on 2×3. Tables 16–18 report the same metrics on 3×3 and 4×4.

Episode Return (Table 11 together with the main-text native return table). PLATO achieves statistically significant best performance in Setups 1–2 on 2×3, while DeepSets is significantly best in Setup 3 (79.25 vs. 76.07); DICG, DGN, and MOHITO lag substantially under task and agent

	2×3			
Model	S0	S1	S2	S3
PLATO	17.26±19.15	**64.29±17.82**	64.82±4.45	56.25±2.30
DICG	11.53±19.80	82.86±14.28	64.82±3.05	57.91±2.48
DGN	37.55±23.75	81.59±13.89	65.69±3.83	58.17±3.33
MOHITO	55.56±9.10	89.81±9.23	67.50±17.62	60.46±6.10
DeepSets	34.94±20.63	93.51±6.26	68.08±4.42	55.12±2.49
NOOP	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
Random	37.92±24.17	94.83±7.96	63.25±9.93	63.97±2.39
Weakest	3.56±8.23	98.57±0.99	60.00±0.00	50.67±0.00
Strongest	3.56±8.23	97.61±4.04	**44.87±11.81**	50.67±0.00

Table 13: Efficiency: NOOP% (mean ± std), trained and tested on 2×3. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

	2×3			
Model	S0	S1	S2	S3
PLATO	0.00±0.00	0.01±0.08	1.07±0.25	1.01±0.08
DICG	0.00±0.00	0.36±0.48	1.05±0.23	1.00±0.00
DGN	0.02±0.14	0.80±0.54	1.10±0.30	1.01±0.12
MOHITO	0.00±0.00	0.45±0.55	1.87±0.33	1.03±0.18
DeepSets	0.00±0.00	0.30±0.46	1.06±0.24	1.00±0.00
NOOP	2.00±0.00	2.00±0.00	2.00±0.00	2.00±0.00
Random	0.02±0.14	0.62±0.57	1.94±0.24	1.38±0.49
Weakest	0.00±0.00	0.16±0.37	1.00±0.00	1.00±0.00
Strongest	0.00±0.00	0.00±0.00	1.40±0.49	1.00±0.00

Table 14: Performance: Burnouts per episode (mean ± std), trained and tested on 2×3. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

	2×3			
Model	S0	S1	S2	S3
PLATO	2.00±0.00	**26.43±12.80**	**35.65±8.83**	39.73±5.28
DICG	2.00±0.00	9.82±7.36	27.87±6.18	37.40±3.91
DGN	1.98±0.14	8.01±5.49	24.90±7.44	36.54±5.40
MOHITO	2.00±0.00	5.29±3.70	6.03±6.82	31.13±7.15
DeepSets	2.00±0.00	5.24±4.00	34.54±8.73	41.40±3.95
NOOP	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
Random	1.98±0.14	2.80±1.28	4.70±5.53	19.62±8.60
Weakest	2.00±0.00	2.00±0.00	34.00±0.00	34.00±0.00
Strongest	2.00±0.00	2.62±1.84	13.14±12.60	33.54±0.50

Table 15: Performance: Putouts per episode (mean ± std), trained and tested on 2×3. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

openness. On 3×3, PLATO achieves the highest returns with statistical significance in Setups 2–3. On the 4×4 grid, PLATO achieves statistically significant highest returns in Setups 1–3, demonstrating its advantage when task diversity and agent openness co-occur. In Setup 0, all smart methods achieve near-equivalent returns across grids, with no significant best.

Burnouts (Tables 14 and 17). On 2×3, no method achieves a statistically significant minimum in any setup; DGN and MOHITO incur the most burnouts in Setup 1 (0.80, 0.45 vs. PLATO 0.01), and MOHITO also in Setup 2 (1.87). On 3×3, PLATO achieves statistically significant fewest burnouts in Setups 2–3. On 4×4, PLATO achieves statistically significant fewest burnouts in Setups 1–3,

	3×3				4×4			
Model	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	30.39±10.46	94.38±2.78	49.15±3.22	57.76±7.07	37.31±11.69	90.62±5.36	47.42±3.41	**62.22±5.14**
DICG	53.96±12.54	93.36±2.99	55.52±6.65	62.17±5.49	72.51±11.87	95.95±1.61	57.67±5.21	69.73±4.95
DGN	45.43±8.54	94.17±3.68	60.94±5.95	65.35±4.70	72.53±6.12	94.83±2.76	67.97±5.31	73.00±4.37
MOHITO	**7.41±10.51**	94.37±3.11	59.17±6.93	59.78±6.07	44.44±4.55	94.71±3.21	77.00±5.26	79.86±4.59
DeepSets	20.58±12.03	93.68±3.05	53.45±10.48	65.59±6.95	31.53±11.28	93.86±3.57	52.31±5.44	65.09±5.63
NOOP	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
Random	46.45±15.84	94.33±3.02	70.37±4.43	67.39±7.99	61.09±15.01	94.79±1.98	77.10±3.38	80.07±2.14
Weakest	12.00±11.19	96.36±1.50	52.25±8.51	64.63±0.10	28.67±13.49	94.33±1.95	57.83±3.79	80.50±0.00
Strongest	12.00±11.19	93.24±5.15	51.50±11.60	54.59±13.51	28.67±13.49	92.61±2.89	49.37±2.92	67.66±3.31

Table 16: Efficiency: NOOP% (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

	3×3				4×4			
Model	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	0.00±0.00	0.12±0.36	**2.07±0.25**	**2.62±0.55**	0.00±0.00	**0.39±0.49**	**2.17±0.40**	**3.57±0.58**
DICG	0.00±0.00	0.51±0.63	2.97±0.35	3.19±0.44	0.00±0.00	0.62±0.67	3.77±0.57	4.59±0.65
DGN	0.09±0.29	0.62±0.70	3.09±0.39	3.30±0.47	0.05±0.21	1.11±0.84	4.68±0.52	4.85±0.43
MOHITO	0.00±0.00	1.24±0.91	3.01±0.47	3.11±0.47	0.00±0.00	2.17±0.84	4.97±0.68	4.99±0.57
DeepSets	0.00±0.00	0.30±0.51	2.46±0.54	2.94±0.31	0.00±0.00	0.92±0.70	3.06±0.65	4.14±0.78
NOOP	4.00±0.00	4.00±0.00	4.00±0.00	4.00±0.00	6.00±0.00	6.00±0.00	6.00±0.00	6.00±0.00
Random	0.50±0.61	0.88±0.87	3.74±0.44	3.88±0.33	0.70±0.79	2.64±1.08	5.14±0.40	5.30±0.46
Weakest	0.00±0.00	0.06±0.24	2.50±0.51	3.00±0.00	0.00±0.00	0.82±0.69	4.42±0.57	5.00±0.00
Strongest	0.00±0.00	0.18±0.44	3.28±0.64	3.32±0.59	0.00±0.00	1.88±0.96	3.18±0.48	4.92±0.27

Table 17: Performance: Burnouts per episode (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

	3×3				4×4			
Model	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	4.00±0.00	3.88±0.36	**60.48±6.22**	**44.89±9.46**	6.00±0.00	10.13±4.50	**118.58±10.79**	**81.19±13.19**
DICG	4.00±0.00	3.49±0.63	34.85±6.68	25.60±7.66	6.00±0.00	6.67±1.95	70.08±10.66	44.75±11.32
DGN	3.91±0.29	3.38±0.70	27.39±7.24	21.08±7.99	5.94±0.26	6.53±2.12	48.17±9.98	39.90±9.40
MOHITO	4.00±0.00	2.76±0.91	26.18±9.63	28.73±8.82	6.00±0.00	5.23±2.77	27.92±13.87	24.93±9.54
DeepSets	4.00±0.00	3.70±0.51	45.88±11.14	34.38±6.18	6.00±0.00	8.06±3.42	89.58±12.10	61.34±14.95
NOOP	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
Random	3.50±0.61	3.12±0.87	11.48±7.72	6.32±6.24	5.30±0.79	4.24±1.25	31.96±10.25	26.06±10.86
Weakest	4.00±0.00	3.94±0.24	39.68±10.83	23.72±0.86	6.00±0.00	6.00±0.99	75.10±9.62	32.70±0.46
Strongest	4.00±0.00	3.82±0.44	17.40±13.95	12.28±9.63	6.00±0.00	5.72±1.65	85.80±5.94	48.24±5.76

Table 18: Performance: Putouts per episode (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.00556$).

reflecting more effective task prioritization. In Setup 0, all smart methods achieve near-zero burnouts on all grids.

Putouts (Tables 15 and 18). PLATO achieves statistically significant best task completions in Setups 1–2 on 2×3; in Setup 3 no method is significantly best (DeepSets 41.40 vs. PLATO 39.73). On 3×3, PLATO achieves statistically significant best putouts in Setups 2–3. On 4×4, PLATO achieves statistically significant best putouts in Setups 2–3. MOHITO achieves the fewest putouts among smart methods under open setups at 4×4, consistent with its higher NOOP rates.

	Trained on 2×3								Trained on 3×3							
	3×3				4×4				4×4				5×5			
Model	S0	S1	S2	S3	S0	S1	S2	S3	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	18.43±14.55	**87.04±7.65**	**51.19±5.34**	**53.80±12.92**	24.03±15.80	86.51±8.30	**49.32±4.96**	56.16±8.73	26.71±14.04	90.18±6.05	50.44±9.25	58.87±4.14	28.81±14.54	92.03±3.80	**46.36±7.08**	**56.28±7.37**
DICG	**6.58±12.55**	93.04±2.78	53.99±7.23	67.41±3.89	22.87±27.10	88.20±7.15	52.61±3.83	56.72±4.41	13.73±20.82	90.03±6.28	51.13±3.76	54.93±3.95	17.09±16.61	91.08±3.28	59.92±6.90	64.71±4.49
DGN	48.57±14.00	93.41±3.25	58.38±5.18	66.10±4.33	49.86±12.57	87.51±6.82	59.13±5.61	57.24±4.60	48.29±13.27	91.39±5.04	58.22±3.64	59.94±4.21	36.10±16.07	90.83±3.50	63.75±5.51	67.44±3.63
MOHITO	55.43±18.20	93.54±5.11	64.49±9.46	72.50±6.75	52.54±8.29	92.32±4.89	73.04±8.92	64.10±5.39	39.16±10.77	91.30±6.50	55.21±8.25	**51.56±2.86**	20.83±9.11	92.20±3.30	54.50±9.04	64.17±5.23
DeepSets	29.39±16.28	93.03±3.33	52.59±6.52	59.16±12.15	30.72±11.91	88.70±4.81	52.90±3.41	58.49±3.92	22.65±14.87	90.13±3.98	59.76±3.13	55.73±3.85	22.76±16.55	89.65±3.70	53.61±7.84	61.63±7.29

Table 19: Generalizability: NOOP% (mean ± std); trained on 2×3, tested on 3×3 and 4×4; trained on 3×3, tested on 4×4 and 5×5. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.01$).

	Trained on 2×3								Trained on 3×3							
	3×3				4×4				4×4				5×5			
Model	S0	S1	S2	S3	S0	S1	S2	S3	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	0.01±0.08	**0.74±0.73**	3.97±0.20	4.63±0.48	0.57±0.63	3.63±0.89	6.01±0.12	6.53±0.50	1.09±0.67	3.83±0.92	6.05±0.61	6.01±0.18	2.06±0.74	4.53±1.00	**6.29±0.47**	6.89±0.55
DICG	0.00±0.00	1.64±0.79	**3.87±0.38**	4.33±0.47	0.68±0.50	3.78±0.93	6.01±0.22	6.07±0.25	1.00±0.37	3.93±0.67	6.00±0.12	6.07±0.25	1.69±0.46	5.03±0.89	6.99±0.37	7.33±0.47
DGN	0.73±0.69	1.77±0.61	3.99±0.20	4.25±0.43	2.47±0.91	4.64±0.87	6.16±0.37	6.11±0.31	1.80±0.97	4.66±0.94	6.07±0.25	6.17±0.37	2.33±0.94	5.33±0.94	7.03±0.29	7.34±0.48
MOHITO	1.53±0.72	2.25±0.79	4.40±0.60	4.77±0.42	2.65±0.52	4.77±0.75	6.80±0.40	6.29±0.46	0.70±0.46	5.03±0.70	6.27±0.48	6.03±0.16	1.62±0.49	4.59±0.96	6.80±0.53	7.27±0.49
DeepSets	0.20±0.40	1.10±0.84	3.92±0.27	4.48±0.50	1.26±0.78	4.76±0.74	6.04±0.20	6.06±0.24	0.80±0.73	4.84±0.74	6.00±0.20	6.02±0.14	2.04±0.67	5.04±1.01	6.70±0.51	7.06±0.51

Table 20: Generalizability: Burnouts per episode (mean ± std); trained on 2×3, tested on 3×3 and 4×4; trained on 3×3, tested on 4×4 and 5×5. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.01$).

	Trained on 2×3								Trained on 3×3							
	3×3				4×4				4×4				5×5			
Model	S0	S1	S2	S3	S0	S1	S2	S3	S0	S1	S2	S3	S0	S1	S2	S3
PLATO	4.99±0.08	**4.15±0.74**	**46.35±6.24**	11.02±13.65	6.43±0.63	6.29±4.51	41.06±7.73	17.37±17.82	5.91±0.67	6.45±4.51	33.63±12.27	38.69±7.04	5.94±0.74	5.77±2.64	**50.69±10.43**	**36.09±12.64**
DICG	5.00±0.00	3.36±0.79	36.18±7.47	20.99±8.72	6.31±0.49	**7.87±4.65**	38.53±8.33	34.41±9.03	6.00±0.37	**8.00±4.28**	39.94±6.49	35.75±7.95	6.31±0.46	5.06±1.67	31.74±8.77	22.08±9.19
DGN	4.27±0.69	3.23±0.61	31.91±6.06	21.83±8.88	4.53±0.91	6.75±4.22	28.97±12.54	32.55±10.34	5.20±0.97	4.37±2.09	32.99±9.54	28.13±13.01	5.67±0.94	3.93±1.39	25.18±7.01	18.45±8.73
MOHITO	3.47±0.72	2.73±0.78	18.59±12.95	6.46±6.40	4.35±0.52	3.94±1.82	3.21±4.33	21.90±11.07	6.30±0.46	4.17±2.89	26.61±15.08	37.91±7.51	6.38±0.49	4.38±1.31	31.19±10.67	23.95±10.53
DeepSets	4.80±0.40	3.90±0.84	42.90±4.89	16.74±14.82	5.74±0.78	4.78±2.31	39.84±9.13	36.46±9.88	6.20±0.73	5.58±3.61	40.12±6.04	39.60±7.35	5.96±0.67	4.84±1.81	39.16±10.72	27.98±10.95

Table 21: Generalizability: Putouts per episode (mean ± std); trained on 2×3, tested on 3×3 and 4×4; trained on 3×3, tested on 4×4 and 5×5. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.01$).

NOOP Usage (Tables 13 and 16). PLATO achieves the statistically significant lowest idle rate in Setup 1 on 2×3; Strongest achieves the statistically significant lowest idle rate in Setup 2 (44.87%); no method reaches significance in Setup 0 or Setup 3. On 3×3, MOHITO achieves the statistically significant lowest NOOP in Setup 0 due to its aggressive suppression policy. On 4×4, PLATO achieves the statistically significant lowest idle rate in Setup 3. All methods show elevated NOOP in Setup 1 on 3×3 due to the higher agent-to-fire ratio.

Reward per Fight (Table 12 together with the main-text native reward-per-fight table). On 2×3, Strongest achieves the statistically significant highest reward per fight in Setup 1; no method reaches significance in Setups 0, 2, or 3. On 3×3, PLATO achieves the statistically significant highest reward per fight in Setups 0, 2–3, reflecting efficient suppressant allocation; Weakest achieves it in Setup 1. On 4×4, PLATO leads S2–S3 and MOHITO leads S0, all with statistical significance.

C.7.2 Zero-Shot: Detailed Results

The main-text zero-shot episode-return table reports episode return for zero-shot transfer. Tables 19–21 provide NOOP%, burnouts, and putouts for policies trained on 2×3 (tested on 3×3 and 4×4) and trained on 3×3 (tested on 4×4 and 5×5).

Episode Return. Results are mixed across zero-shot target grids. In the bottom half of the main-text zero-shot episode-return table, policies trained on 3×3 show that PLATO leads S1–S3 at 5×5 with statistical significance, while DICG leads S1 at 4×4. In the top half of that table, policies trained on 2×3 show that PLATO leads S1–S2 at 3×3, PLATO and DeepSets are statistically tied in S2 at 4×4, and DICG leads S1 at 4×4; MOHITO shows negative returns in S3 at 3×3 and S2 at 4×4. DGN degrades consistently under higher openness across all training sources.

Burnouts (Table 20). Trained on 2×3, PLATO achieves statistically significant fewest burnouts in S1 at 3×3; DICG achieves statistically significant fewest in S2 at 3×3. Trained on 3×3, PLATO achieves statistically significant fewest burnouts in S2 at 5×5. All methods accumulate near-maximal burnouts in S2–S3 at 4×4 regardless of training source.

PLATO Variation	2×3			
	S0	S1	S2	S3
(mlp+add)	4.00±0.00	52.84±25.62	68.17±17.80	76.07±10.73
(mlp+dot)	4.00±0.00	39.99±21.20	68.63±13.85	70.84±21.94
(lstm+add)	4.00±0.00	66.95±17.77	70.72±11.14	75.84±9.24
(lstm+dot)	4.00±0.00	33.15±19.01	68.99±14.56	74.93±12.55

Table 22: Ablation: Episode return (mean ± std), trained and tested on 2×3. No variant achieves statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

Putouts (Table 21). Trained on 2×3, PLATO leads S1–S2 at 3×3 with statistical significance; DICG leads S1 at 4×4 and MOHITO collapses in S2 (3.21 putouts). Trained on 3×3, PLATO leads at 5×5 in S2–S3 with statistical significance; DICG leads S1 at 4×4.

NOOP Usage (Table 19). When trained on 2×3, PLATO achieves the statistically significant lowest idle rates in S1–S3 at 3×3 and S2 at 4×4; DICG achieves the statistically significant lowest NOOP in S0 at 3×3. When trained on 3×3, PLATO leads at 5×5 in S2–S3; MOHITO achieves the statistically significant lowest NOOP in S3 at 4×4. All methods show very high NOOP in Setup 1 under both training regimes, reflecting task sparsity in the agent-openness phase.

Zero-Shot Summary. PLATO generalizes most consistently across target grids and metrics, with significantly lowest NOOP in five of eight S2–S3 target settings and significantly most putouts in S2 at 3×3 and S2–S3 at 5×5. DICG competes significantly in putouts at 4×4 from both training sources. MOHITO degrades severely in S2 at 4×4 from 2×3 training (return -13.17, 3.21 putouts). Training on 3×3 generally yields stronger transfer to 4×4 than 2×3 in S2–S3, and more stable transfer in S3.

C.8 Ablation Studies: Detailed Results

This section provides detailed NOOP%, burnout, putout, and reward-per-fight results for the PLATO ablation study. The main-text ablation episode-return table reports episode return for both native and zero-shot settings.

C.8.1 Variant Descriptions

We ablate two design choices:

MLP vs. LSTM encoder-decoder. *The actor encodes each observation independently, with no memory of prior timesteps.* In the MLP variant (our primary), a feedforward network maps the agent's current observation directly to a query vector. The LSTM variant replaces this with a recurrent unit that maintains a hidden state across timesteps, allowing the actor to condition task selection on the history of observations within an episode. Both variants use the same additive attention scorer unless otherwise stated.

Additive vs. dot-product attention scorer. *The pointer scores agent–task compatibility via inner product rather than learned projections.* The additive (Bahdanau) scorer applies learned projection matrices W_q, W_K, and a learned vector v to compute $u_x = v^\top \tanh(W_K k_x + W_q q)$, capturing asymmetric relationships at the cost of extra parameters. The dot-product scorer computes $u_x = q^\top k_x$ (after projection to a shared space), using direct inner-product similarity with no additional parameters. Both scorer variants are evaluated within the MLP and LSTM encoder settings.

C.8.2 Detailed Native Results

Table 22 reports episode return on 2×3. Tables 23–29 report NOOP%, burnouts, putouts, and reward per fight for the 2×3 native setting. Tables 24–30 report the corresponding native metrics on 3×3 and 4×4.

PLATO Variation	2 × 3			
	S0	S1	S2	S3
(mlp+add)	17.26±19.15	64.29±17.82	**64.82±4.45**	56.25±2.30
(mlp+dot)	26.98±22.99	65.24±17.11	67.61±2.16	57.12±3.05
(lstm+add)	14.04±17.09	57.97±10.03	68.31±2.42	56.53±2.49
(lstm+dot)	19.76±19.45	80.26±13.01	68.47±1.96	56.34±2.54

Table 23: Ablation: NOOP% (mean ± std), trained and tested on 2×3. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

PLATO Variation	3 × 3				4 × 4			
	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	30.39±10.46	**94.38±2.78**	49.15±3.22	57.76±7.07	37.31±11.69	90.62±5.36	47.42±3.41	62.22±5.14
(mlp+dot)	20.16±10.90	95.61±2.00	46.64±3.49	57.03±7.04	34.73±9.90	90.46±5.30	47.14±3.25	61.88±5.30
(lstm+add)	**13.48±12.17**	94.92±2.75	49.78±3.82	55.81±6.66	40.18±11.71	94.99±3.69	47.27±3.43	63.46±4.25
(lstm+dot)	35.84±13.21	95.10±3.19	48.62±4.35	56.90±7.24	34.12±17.32	91.94±4.64	46.73±3.24	62.44±4.50

Table 24: Ablation: NOOP% (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

PLATO Variation	2 × 3			
	S0	S1	S2	S3
(mlp+add)	0.00±0.00	0.01±0.08	1.07±0.25	1.01±0.08
(mlp+dot)	0.00±0.00	0.24±0.44	1.03±0.18	1.07±0.26
(lstm+add)	0.00±0.00	0.03±0.18	1.02±0.14	1.00±0.00
(lstm+dot)	0.00±0.00	0.02±0.14	1.04±0.20	1.01±0.12

Table 25: Ablation: Burnouts per episode (mean ± std), trained and tested on 2×3. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

PLATO Variation	3 × 3				4 × 4			
	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	0.00±0.00	0.12±0.36	2.07±0.25	2.62±0.55	0.00±0.00	0.39±0.49	2.17±0.40	3.57±0.58
(mlp+dot)	0.00±0.00	0.18±0.39	2.09±0.29	2.58±0.55	0.00±0.00	0.36±0.48	2.23±0.47	3.51±0.62
(lstm+add)	0.00±0.00	0.13±0.34	2.09±0.28	2.52±0.58	0.00±0.00	**0.17±0.38**	2.21±0.45	3.40±0.54
(lstm+dot)	0.00±0.00	0.20±0.40	2.11±0.31	2.47±0.51	0.00±0.00	0.32±0.48	2.25±0.48	3.48±0.60

Table 26: Ablation: Burnouts per episode (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

Burnouts (Tables 25 and 26). Burnout rates are similar across all variants. All variants achieve zero burnouts in Setup 0. PLATO (lstm+add) achieves significantly fewest burnouts in Setup 1 at 4×4 (0.17).

Putouts (Tables 27 and 28). Additive variants generally achieve more putouts than dot-product variants in Setup 1, with (lstm+add) achieving the highest putouts in S1 at 2×3 (33.51) among all variants. No variant achieves statistically significant best performance in any setup.

NOOP Usage (Tables 23 and 24). On 2×3 S2, mlp+add achieves the statistically significant lowest NOOP (64.82%). On 3×3 S0, lstm+add achieves statistically significant lowest NOOP

PLATO Variation	2 × 3			
	S0	S1	S2	S3
(mlp+add)	2.00±0.00	26.43±12.80	35.65±8.83	39.73±5.28
(mlp+dot)	2.00±0.00	20.23±10.42	36.31±6.75	37.15±10.69
(lstm+add)	2.00±0.00	33.51±8.84	37.32±5.45	39.59±4.57
(lstm+dot)	2.00±0.00	16.59±9.52	36.52±7.11	39.21±6.13

Table 27: Ablation: Putouts per episode (mean ± std), trained and tested on 2×3. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

PLATO Variation	3 × 3				4 × 4			
	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	4.00±0.00	3.88±0.36	60.48±6.22	44.89±9.46	6.00±0.00	10.13±4.50	118.58±10.79	81.19±13.19
(mlp+dot)	4.00±0.00	3.82±0.39	58.92±6.67	46.77±10.00	6.00±0.00	10.24±4.60	114.75±11.71	79.81±12.51
(lstm+add)	4.00±0.00	3.87±0.34	58.37±7.06	45.58±10.75	6.00±0.00	8.23±3.32	118.00±11.99	82.07±11.81
(lstm+dot)	4.00±0.00	3.80±0.40	60.94±7.47	48.55±8.57	6.00±0.00	10.07±4.16	113.65±11.42	80.39±12.72

Table 28: Ablation: Putouts per episode (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

PLATO Variation	2 × 3			
	S0	S1	S2	S3
(mlp+add)	1.27±0.45	0.52±0.12	0.67±0.17	0.58±0.10
(mlp+dot)	1.29±0.49	0.40±0.11	0.72±0.12	0.53±0.24
(lstm+add)	1.29±0.44	0.54±0.09	0.75±0.11	0.58±0.07
(lstm+dot)	1.29±0.48	**0.62±0.19**	0.73±0.17	0.57±0.12

Table 29: Ablation: Reward per fight (mean ± std), trained and tested on 2×3. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

PLATO Variation	3 × 3				4 × 4			
	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	1.25±0.23	0.60±0.38	0.74±0.08	0.64±0.10	1.09±0.25	0.41±0.21	0.73±0.06	0.66±0.07
(mlp+dot)	1.11±0.15	0.70±0.39	0.69±0.08	0.66±0.11	1.06±0.22	0.40±0.19	0.70±0.06	0.64±0.08
(lstm+add)	1.04±0.16	0.64±0.37	0.72±0.09	0.62±0.13	1.07±0.24	**0.72±0.35**	0.72±0.06	0.69±0.08
(lstm+dot)	1.27±0.30	0.67±0.41	0.74±0.09	0.68±0.09	1.26±0.35	0.49±0.25	0.68±0.06	0.66±0.07

Table 30: Ablation: Reward per fight (mean ± std), trained and tested on 3×3 and 4×4. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

(13.48%); mlp+add achieves significantly lowest NOOP in 3×3 S1 (94.38%), though all variants remain above 94%, reflecting task sparsity.

Reward per Fight (Tables 29 and 30). Per-fight efficiency is similar across all variants. PLATO (lstm+dot) achieves statistically significant highest per-fight efficiency in 2×3 S1 (0.62). PLATO (lstm+add) leads in 4×4 S1 (0.72, significant).

	Trained on 2×3							
	3×3				4×4			
PLATO Variation	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	18.43±14.55	87.04±7.65	51.19±5.34	53.80±12.92	24.03±15.80	86.51±8.30	49.32±4.96	56.16±8.73
(mlp+dot)	16.01±12.32	86.31±7.93	53.17±6.62	54.31±12.95	**17.49±16.76**	83.86±9.84	47.80±4.28	55.50±7.80
(lstm+add)	21.81±17.80	83.29±9.11	55.54±7.83	60.18±11.12	24.18±13.96	83.05±12.68	49.32±3.63	58.42±6.69
(lstm+dot)	17.59±16.86	84.06±10.97	**48.17±5.42**	54.83±12.86	25.18±16.46	82.39±11.17	50.66±4.95	55.07±7.69

Table 31: Ablation: Zero-shot NOOP% (mean ± std), trained on 2×3, tested on 3×3 and 4×4. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

	Trained on 3×3							
	4×4				5×5			
PLATO Variation	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	26.71±14.04	90.18±6.05	50.44±9.25	58.87±4.14	28.81±14.54	92.03±3.80	46.36±7.08	56.28±7.37
(mlp+dot)	36.33±20.78	91.28±3.99	49.82±9.14	53.23±3.30	22.62±14.49	91.60±3.95	46.09±7.19	58.09±7.07
(lstm+add)	**20.94±14.18**	**79.13±15.81**	50.79±9.95	53.46±3.83	19.66±13.38	90.93±3.71	**45.45±6.54**	54.66±6.88
(lstm+dot)	25.68±16.11	91.05±3.92	55.97±8.70	53.08±2.95	26.99±13.88	**89.03±6.16**	45.53±5.71	55.95±7.47

Table 32: Ablation: Zero-shot NOOP% (mean ± std), trained on 3×3, tested on 4×4 and 5×5. Bold indicates statistically significant minimum (lowest idle rate) within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

	Trained on 2×3							
	3×3				4×4			
PLATO Variation	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	0.01±0.08	0.74±0.73	3.97±0.20	4.63±0.48	0.57±0.63	3.63±0.89	6.01±0.12	6.53±0.50
(mlp+dot)	0.00±0.00	0.92±0.76	4.00±0.00	4.63±0.49	**0.26±0.48**	3.79±0.89	6.02±0.14	6.47±0.50
(lstm+add)	0.04±0.23	0.63±0.69	4.00±0.48	4.45±0.50	0.73±0.69	4.08±0.79	6.01±0.08	**6.23±0.42**
(lstm+dot)	0.03±0.20	0.83±0.86	**3.77±0.45**	4.61±0.49	0.92±0.74	3.57±0.81	6.03±0.21	6.36±0.50

Table 33: Ablation: Zero-shot burnouts per episode (mean ± std), trained on 2×3, tested on 3×3 and 4×4. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

	Trained on 3×3							
	4×4				5×5			
PLATO Variation	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	1.09±0.67	3.83±0.92	6.05±0.61	6.01±0.18	2.06±0.74	4.53±1.00	6.29±0.47	6.89±0.55
(mlp+dot)	0.96±0.57	3.63±0.89	6.05±0.58	6.01±0.14	**1.62±0.71**	**4.13±0.85**	6.29±0.48	7.06±0.49
(lstm+add)	0.99±0.73	4.40±0.94	6.15±0.57	6.03±0.21	1.99±0.69	4.75±1.07	6.25±0.44	6.77±0.61
(lstm+dot)	1.03±0.67	4.06±0.77	6.28±0.68	5.99±0.08	2.07±0.77	5.24±0.79	6.19±0.39	6.71±0.59

Table 34: Ablation: Zero-shot burnouts per episode (mean ± std), trained on 3×3, tested on 4×4 and 5×5. Bold indicates statistically significant minimum within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

C.8.3 Detailed Zero-Shot Results

Tables 31–35 report the zero-shot NOOP%, burnouts, and putouts for policies trained on 2×3 and tested on 3×3 and 4×4. Tables 32–36 report the corresponding zero-shot metrics for policies trained on 3×3 and tested on 4×4 and 5×5.

	Trained on 2×3							
	3×3				4×4			
PLATO Variation	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	4.99±0.08	4.15±0.74	46.35±6.24	11.02±13.65	6.43±0.63	6.29±4.51	41.06±7.73	17.37±17.82
(mlp+dot)	5.00±0.00	4.05±0.76	45.09±6.01	10.79±13.33	**6.74±0.48**	5.83±3.83	40.61±8.12	19.39±17.95
(lstm+add)	4.96±0.23	3.89±0.72	41.89±10.80	**17.61±13.37**	6.27±0.69	**7.77±4.56**	42.43±5.79	**29.22±15.87**
(lstm+dot)	4.97±0.20	3.99±0.87	47.21±8.08	11.64±13.83	6.08±0.74	5.85±2.83	41.91±8.93	23.13±17.72

Table 35: Ablation: Zero-shot putouts per episode (mean ± std), trained on 2×3, tested on 3×3 and 4×4. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

	Trained on 3×3							
	4×4				5×5			
PLATO Variation	S0	S1	S2	S3	S0	S1	S2	S3
(mlp+add)	5.91±0.67	6.45±4.51	33.63±12.27	38.69±7.04	5.94±0.74	5.77±2.64	50.69±10.43	36.09±12.64
(mlp+dot)	6.04±0.57	5.61±2.32	33.84±12.79	41.53±6.41	**6.38±0.71**	5.13±1.41	48.23±9.81	31.28±11.02
(lstm+add)	6.01±0.73	**8.39±5.27**	29.99±13.09	39.35±8.70	6.01±0.69	5.40±2.34	50.95±10.61	38.76±12.87
(lstm+dot)	5.97±0.67	5.71±2.23	28.13±13.99	41.15±4.99	5.93±0.77	7.21±3.36	**54.33±9.87**	37.95±12.36

Table 36: Ablation: Zero-shot putouts per episode (mean ± std), trained on 3×3, tested on 4×4 and 5×5. Bold indicates statistically significant best performance within each setup (Shapiro-Wilk-adaptive two-sided test; Bonferroni-corrected, $p<0.0125$).

Zero-Shot Burnouts (Tables 33 and 34). Burnout counts are near-maximal across variants in S2–S3 at 4×4 (≈ 6.0) regardless of training source, reflecting the difficulty of fire control at scale. Trained on 2×3, mlp+dot achieves statistically significant fewest burnouts in S0 at 4×4 (0.26); lstm+dot achieves statistically significant fewest in S2 at 3×3 (3.77); lstm+add achieves statistically significant fewest in S3 at 4×4 (6.23). Trained on 3×3, mlp+dot achieves statistically significant fewest burnouts in S0–S1 at 5×5.

Zero-Shot Putouts (Tables 35 and 36). Trained on 2×3, mlp+dot achieves statistically significant best putouts in S0 at 4×4 (6.74); lstm+add achieves statistically significant best putouts in S3 at 3×3 (17.61) and S1 and S3 at 4×4. Trained on 3×3, mlp+dot achieves statistically significant best putouts in S0 at 5×5 (6.38); lstm+add achieves significantly most putouts at 4×4 S1 (8.39); lstm+dot achieves best putouts at 5×5 S2 (54.33, significant).

Zero-Shot NOOP Usage (Tables 31 and 32). NOOP rates are high in Setup 1 (79%–92%) across all variants and training sources, reflecting task sparsity. Trained on 2×3, mlp+dot achieves statistically significant lowest NOOP in S0 at 4×4 (17.49%), and lstm+dot achieves lowest in S2 at 3×3 (48.17%). Trained on 3×3, lstm+add achieves significantly lowest NOOP in S0–S1 at 4×4 and S2 at 5×5; lstm+dot achieves significantly lowest NOOP in S1 at 5×5 (89.03%).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state four contributions: (1) PLATO as the first MARL method addressing AO and TO jointly without fixed bounds or retraining, (2) the TaAgO-MG formalization with formal well-definedness and permutation-invariance proofs (Appendix A), (3) empirical comparison against baselines across openness setups (Sections 5–6), and (4) zero-shot generalization to unseen grid sizes. All claims are directly supported by the experimental results and formal proofs presented in the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion (Section 8) discusses three limitations: (1) the current TaAgO-MG formulation assumes one action per task, restricting application to domains requiring per-task action sets; (2) the policy assumes full observability of agents and tasks, which may not hold in practice; and (3) evaluation is confined to the MOASEI wildfire domain, leaving generalization to other open-system domains as future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We include a lemma and theorems outlining our proofs in the main body of the paper, with full proofs and assumptions in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include extensive reproducibility details in Appendix B. This includes both the hyperparameter tuning space and the final hyperparameters for shown results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will publish the code and reasonable documentation used to produce our experiments upon publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe enough of the Wildfire environment to understand the results shown in the main body of the paper. We formally define reward bounds for Wildfire in A.6, and we provide additional explanation of the domain in C.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We show the std in all tables as well as denoting Shapiro-Wilk-adaptive two-sided significance (with Bonferroni correction) in bold.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In Appendix B we include hardware details, hyperparameter tuning ranges, and compute estimates. Each training run takes approximately 3–5 CPU-days; the full suite (all methods, grids, setups, and tuning trials) required approximately 6,000–10,000 CPU-days in total.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We reviewed the listed guidelines and ensured our submission adheres to them.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Openness work remains fairly theoretical for now, and is thus detached from societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our method has no known or anticipated risks with release.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: This work uses the free-range-zoo API for its experiments. This is under the GNU3 license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: Other than the code upon publication, we do not include any additional assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: Our work does not include any human studies or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Our work does not include any human studies or crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Our work does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.