Auction-Based Online Policy Adaptation for Evolving Objectives

Guruprerana Shabadi
University of Pennsylvania
United States
shabadi@engineering.upenn.edu

Kaushik Mallik
IMDEA Software Institute
Spain
kaushik.mallik@imdea.org

Abstract

We consider multi-objective reinforcement learning problems where objectives come from an identical family—such as the class of reachability objectives—and may appear or disappear at runtime. Our goal is to design adaptive policies that can efficiently adjust their behaviors as the set of active objectives changes. To solve this problem, we propose a modular framework where each objective is supported by a selfish local policy, and coordination is achieved through a novel *auction*-based mechanism: policies bid for the right to execute their actions, with bids reflecting the urgency of the current state. The highest bidder selects the action, enabling a dynamic and interpretable trade-off among objectives. Going back to the original adaptation problem, when objectives change, the system adapts by simply adding or removing the corresponding policies. Moreover, as objectives arise from the same family, identical copies of a parameterized policy can be deployed, facilitating immediate adaptation at runtime. We show how the selfish local policies can be computed by turning the problem into a general-sum Markov game, where the policies compete against each other to fulfill their own objectives. To succeed, each policy must not only optimize its own objective, but also reason about the presence of other goals and learn to produce calibrated bids that reflect relative priority. Under mild assumptions, we prove the existence of Nash equilibria where dishonest bidding leads to suboptimal outcome, and the most urgent objectives win control automatically. In our implementation, the policies are trained concurrently using proximal policy optimization (PPO). We evaluate on two Atari games and a gridworld-based path-planning task with dynamic targets. Our method achieves substantially better performance than monolithic policies trained with PPO.

1 Introduction

Consider the problem of controlling a mobile robot deployed in a cat shelter whose task is to deliver food to cats roaming around the facility (Figure 1). New food requests from hungry cats may arrive at arbitrary times and locations, while existing requests may disappear if a cat wanders away before being served. Since the robot can serve only one cat at a time, fulfilling one request may delay others, potentially allowing some cats to move farther away. Thus, the robot must dynamically adapt to the set of active requests to maximize the number of cats fed before they disappear.

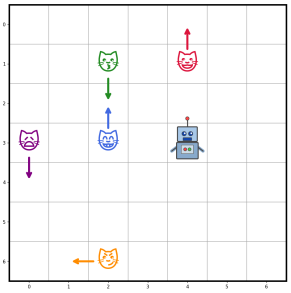


Figure 1: Cat Feeder env.

In this work, we model the above control problem as a multi-objective reinforcement learning task in which all the objectives come from an identical family, such as reachability in this case.

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

These objectives can be conflicting and any number of them may be active at a given time. Moreover, we make no assumptions about their distribution or arrival pattern. Our goal is to design adaptive policies that update their behavior seamlessly as objectives appear or disappear.

The class of problems we study is relevant to many real-world scenarios. For example, a warehouse robot must be responsive to changing pickup, delivery, and sorting tasks, and this setting has motivated work on warehouse robot path planning and resource allocation [9, 26]. More broadly, such problems have been extensively studied in the planning literature [12], typically assuming that a precise environment model (like a graph) is available without uncertainties. Existing reinforcement learning-based path planning methods [29] are able to handle uncertainties but cannot adapt to varying number of objectives at deployment time.

To solve this problem, we propose a novel compositional RL framework, where each objective is served using a local policy, selfishly maximizing its assigned objective by competing against the other policies. Coordination between these policies is achieved through a novel auction-based mechanism: at regular intervals of a given length $\tau > 0$, policies submit bids for the right to execute their actions, where bids reflect the urgency of the current state with respect to their objectives. The highest bidder selects the action for the next τ steps, enabling a dynamic and interpretable trade-off among objectives. Returning to the cat feeder example, suppose the robot approaches an intersection with two active requests, the policy for the first request recommends turning left, while the policy for the second recommends turning right. Since both actions cannot be executed simultaneously, a trade-off is unavoidable. The auction mechanism determines which objective is prioritized, based on the relative urgency encoded in the bids.

This compositional design yields modularity, facilitating fast adaptation: when an objective appears or disappears, only the corresponding policy needs to be added or removed. Furthermore, when the objectives belong to the identical family—which is reachability in the cat feeder example—a single generalized policy can be trained for the entire family. We can deploy identical *copies* of this policy when new objectives arrive, enabling immediate adaptation. Modularity also permits us to define objective-specific reward signals while the auction mechanism takes care of the trade-offs between objectives. For example, we can easily define a distance-based dense reward signal for each policy. In contrast, in the monolithic case, it is challenging to define such a dense reward signal that strikes the "optimal" trade-off between all objectives. We show in our experiments that monolithic policies, trained with both sparse rewards and different types of scalarized dense rewards, achieves significantly lower payoffs as compared to our modular approach.

Yet another advantage of our modular framework is interpretability. At any moment, we can observe the policy in control, and thereby the objective currently being pursued by the agent. This explanation is beneficial in a range of real-world situations, like providing post-hoc explanations of failures, and also feedback on biases in pursuing different objectives.

We implement our modular framework as a general-sum Markov game between local policies seeking to fulfill their assigned objectives, where each policy's action space is augmented with an additional numeric variable representing the bids. However, new challenges arise, and below we summarize how we overcome them.

Challenge I: Enforcing honest bids. The success of the auction mechanism hinges on the policies faithfully selecting their bid values according to their true urgency. But what if policies learn to overbid to suppress competitors? This would lead to suboptimal results. To prevent this, policies are penalized in proportion to their bid values. As a consequence, excessive bidding reduces the net return of a policy, which is the total accumulated rewards minus the total accumulated bidding penalties. Under mild assumptions, we prove that our penalty model implements truthful bidding, i.e., policies would bid high only if the earned rewards outweigh the penalties.

Challenge II: Achieving environment awareness. Effective bidding not only depends on the current state and the local objective, but also on the objectives of the competing policies. For instance, in the cat feeder problem, if two requests are spatially close, their policies need not compete aggressively. Conversely, if requests lie in opposite directions, the urgent policy must bid sufficiently high to win. To enable such environment-aware bidding, we allow the policies to observe all the objectives, i.e., the policies have information about the global state. But a large number of objectives makes the global state vector very big, resulting in sluggish training. We equip policies with an *attention pooling* architecture that can extract relevant information from the active objectives into a succinct embedding

vector, so that it can be fed as input to the policy. Each local policy is equipped with its own pooling module which is co-trained with the policy.

Challenge III: Variable objective count. Recall, we set out to support environments where objectives appear and disappear freely at runtime. This feature is automatically incorporated by the attention pooling module, which is a learned component with the ability to generalize against different number of competing objectives than the number seen at training time. Regardless of the objective count, the learned policy always receives a fixed-size encoding of the global state of the environment.

We test our framework on three environments with dynamic expected total payoff objectives: the cat feeder environment implemented as a gridworld, and two Atari games: Assault and Air-raid [6]. We use proximal policy optimization (PPO; [25]) to train each individual policy competing against other policies in the Markov game encoding of the problem. We report the empirical convergence of the training process, and show that policies trained using our framework exhibit dramatic improvement over the baselines in terms of performance. We also demonstrate the ability of our approach to generalize to more objectives at test-time than during training. Lastly, to aid interpretability in renderings of rollouts, we implement visualization of the policy in control and the objective being pursued, through which we get real-time feedback on the strategic behaviors of the policies (videos are included in the Supplementary Material).

Related Work. Most multi-objective reinforcement learning (MORL) methods reduce multiple rewards to a single decision rule, either through scalarization [13, 31], Pareto-based criteria [22, 30], or fairness objectives [8, 21, 27]. These approaches typically learn a monolithic policy for a fixed set of reward components. In contrast, our method keeps objectives modular: each local policy pursues one objective, and the auction mechanism composes their actions at runtime.

The closest distributed MORL approaches are W-learning and Deep W-learning [15, 23], which train separate policies and select among them using learned urgency-like scores. Other decomposition methods combine action values or votes through fixed aggregation rules [19, 24]. Our approach differs by using bids as the coordination interface, which lets standard policy-gradient training learn both the local action and the urgency signal without a separate meta-policy.

Our design is also related to hierarchical RL, which uses modular subpolicies under a higher-level manager [16]. In comparison, our composition is flat and competitive rather than manager-driven. Another related line of work is using auction mechanisms for model-based multi-objective planning and bidding games [2–4, 17]. In comparison, our setting uses model-free RL and supports a varying number of objectives.

2 Auction-Based Multi-Policy Framework

MO-MDPs, Policies, and Values. We use multi-objective Markov decision processes to model uncertain environments with multiple reward objectives. An MO-MDP with $m \in \mathbb{N}_{>0}$ objectives is a tuple $\mathcal{M} = (S, A, T, \mu, R, H)$, where S is the *state* space, A is the *action* space, $T\colon S \times A \to \Delta(S)$ is the *transition* kernel, $\mu \in \Delta(S)$ is the *initial state* distribution, $R = \{r_i\colon S \times A \to \mathbb{R}\}_{i\in[1;m]}$ is the set of *reward functions*, and $H \in \mathbb{N}_{>0}$ is the *time horizon*. A (Markov) *policy* $\pi : S \to \Delta(A)$ maps the current state to a distribution over actions. For a trace $\rho = s^0a^0s^1a^1 \dots s^H$, the accumulated *local* reward of objective i is $w_i(\rho) = \sum_{t=0}^{H-1} r_i(s^t, a^t)$. In the environments we consider, the optimization metric is the cumulative reward across objectives and so we define the *global* reward (aka, scalarized reward [31]) as $w(\rho) = \sum_{i=1}^{m} w_i(\rho)$. The *value* of a policy is $V^\pi = \mathbb{E}^\pi[w(\rho)]$, and an *optimal* policy is any policy $\pi^* \in \arg\max_\pi V^\pi$.

We present a novel *auction*-based RL framework for compositional policy synthesis for MO-MDPs. In our framework, we have objective-specific local policies that not only emit actions, but also submit *bids* within a given bound $\beta \in \mathbb{N}_{>0}$ to win the privilege of executing their actions for a given number of time steps $\tau \in \mathbb{N}_{>0}$. We consider two auction mechanisms, called Winner-Pays and All-Pay. In Winner-Pays, the bids are non-negative *integers* in the interval $[0; \beta]$, and in All-Pay, the bids are non-negative *real* numbers in the interval $[0, \beta]$.[1] In both settings, the highest bidder's actions get executed for τ consecutive steps, with bidding ties being resolved uniformly at random. To encourage

[1]The distinction in bid types—*integers* for Winner-Pays and *reals* for All-Pay—is purely technical. The synthesis of local policies will involve a game-theoretic analysis involving Nash equilibria. For real-valued bids in Winner-Pays, it is unclear if Nash equilibria would always exist, owing to the continuous bid values and discontinuities in the payoff profile due to the

truthfulness, we penalize the policies proportional to their bids scaled with a constant $\rho \in (0, 1)$. In the Winner-Pays model, only the executed policy (one of the highest bidders) pays the penalty, whereas in All-Pay, all policies pay penalties. The penalties force policies to truthfully select their bids to reflect how critical their action actually is in the current state. We remark that this framework is a purely decentralized scheme to coordinate local policies in a given MO-MDP. The parameters τ, β, and ρ denote the *bidding interval*, *maximum bid*, and *bid penalty coefficient*, respectively. Ablation studies in Section 4 highlight their importance. If τ is too small, policies switch too frequently, risking suboptimal outcomes. If β is too small, policies cannot express preference strength, leading to near-random control. If ρ is too large, policies tend to abstain from participating in the bidding.

We formalize our framework as a general-sum game between multiple policies.

Definition 1 (Markov bidding game)**.** Let $\mathcal{M} = (S, A, T, \mu, R, H)$ be an MO-MDP with m objectives, let $\tau \in \mathbb{N}_{>0}$ be the bidding interval, let $\beta \in \mathbb{R}_{>0}$ be the maximum bid, let $\rho \in (0, 1)$ be the bid penalty coefficient, and let $\mathsf{P} \in \{\textsf{Winner-Pays}, \textsf{All-Pay}\}$ be the penalty model. The induced Markov bidding game between the m local policies is $\mathcal{G}^{\mathcal{M}}_{\tau,\beta,\rho,\mathsf{P}} = (\hat{S}, \hat{A}, \hat{T}, \hat{R}, \hat{\mu}, H)$, where:

- $\hat{S} = S \times [1; m] \times [0; \tau - 1]$. A state (s, k, q) records the MO-MDP state s, the current controlling policy k, and the number q of steps remaining before the next bidding round. When $q = 0$, a new bidding round occurs and the current controller k is ignored.
- $\hat{A} = (A \times [0; \beta])^m$ for $\mathsf{P} = \textsf{Winner-Pays}$ and $\hat{A} = (A \times [0, \beta]^m)$ for $\mathsf{P} = \textsf{All-Pay}$. Each policy i independently chooses an action $a_i \in A$ and a bid b_i, where $b_i \in [0; \beta]$ or $b_i \in [0, \beta]$ depending on P. We write a joint action as $(\mathbf{a}, \mathbf{b}) = (a_i, b_i)_{i \in [1;m]}$.
- Let $W(\mathbf{b}) = \{i \in [1; m] \mid b_i = \max_j b_j\}$ be the set of highest bidders. The transition kernel $\hat{T} : \hat{S} \times \hat{A} \to \Delta(\hat{S})$ is defined by

$$\hat{T}((s,k,q),(\mathbf{a},\mathbf{b}),(s',k',q')) = \begin{cases} T(s, a_k, s') & q > 0 \wedge k' = k \wedge q' = q - 1, \\ \frac{1}{|W(\mathbf{b})|} T(s, a_{k'}, s') & q = 0 \wedge k' \in W(\mathbf{b}) \wedge q' = \tau - 1, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

  Intuitively, the case "$q > 0$" means it is not yet a bidding round, and the currently active policy k continues to be used, and the case "$q = 0$" means it is a bidding round, and a new policy is selected via uniform tie-breaking over the set W.
- For each policy i, the reward is inherited from the MO-MDP reward r_i, with a bid penalty charged at bidding rounds:

$$\hat{r}_i((s,k,q),(\mathbf{a},\mathbf{b}),(s',k',q')) = r_i(s, a_{k'}) - \rho b_i \cdot \mathbf{1}[q = 0 \wedge (i = k' \vee \mathsf{P} = \textsf{All-Pay})]\,. \tag{2}$$

- The initial distribution $\hat{\mu}$ satisfies $\hat{\mu}(s, k, 0) = \mu(s)/m$ for every $k \in [1; m]$, and $\hat{\mu}(s, k, q) = 0$ for $q > 0$.

Thus a local policy i in the MO-MDP $\mathcal{M}$ is a strategy $\pi_i \colon \hat{S} \to \hat{A}_i$ in the Markov bidding game $\mathcal{G}^{\mathcal{M}}_{\tau,\beta,\rho,\mathsf{P}}$ maximizing its own expected cumulative values of $\mathbb{E}[\sum_{t=0}^{H-1} \hat{r}_i^t]$, against every other strategy π_j that is maximizing its own expected cumulative reward $\mathbb{E}[\sum_{t=0}^{H-1} \hat{r}_j^t]$, while the auction rule determines which local action is executed. We write π_i as a pair (π_i^a, π_i^b), where $\pi_i^a \colon \hat{S} \to A$ is the action component and $\pi_i^b \colon \hat{S} \to [0; \beta]$ is the bid component for Winner-Pays and $\pi_i^b \colon \hat{S} \to [0, \beta]$ is the same for All-Pay. We next analyze the bidding incentives at a single decision state, using continuation values from this game.

Equilibrium continuation values. We consider local policies $\pi_i = (\pi_i^a, \pi_i^b)$ that form a Nash equilibrium of the Markov bidding game $\mathcal{G}^{\mathcal{M}}_{\tau,\beta,\rho,\mathsf{P}}$. For simplicity, assume H is a multiple of τ (otherwise, extend the horizon to the next multiple of τ). The policies bid at *bidding rounds* $t \in \{0, \tau, 2\tau, \ldots, H - \tau\}$. Given a policy index $i \in [1; m]$, time $t \leq H$, and state $s \in \hat{S}$ of $\mathcal{G}^{\mathcal{M}}_{\tau,\beta,\rho,\mathsf{P}}$, we write $V^*_{i,t}$ to denote the value policy i achieves in the rest of the game starting at time t and at state

tie-breaking mechanism [28]. For integer-valued bids in All-Pay, the shape of the Nash equilibria policies is only known for the two-policy case [11], but the general case is still open. We leave these unresolved cases for a future work.

s, when all the policies follow a Nash equilibrium. Since the horizon H is finite, we can define $V^*_{i,t}$ using backward induction over t starting from $t = H$. Set $V^*_{i,H}(s) = 0$ for every i and s. Suppose the game is at state s and bidding round $t \in \{0, \tau, 2\tau, \ldots, H - \tau\}$.

The *winning continuation value* of a policy i is the optimal value it can achieve upon winning bidding:

$$\widetilde{V}^{\mathsf{win}}_{i,t}(s) \coloneqq \max_{\pi^a_i} \mathbb{E}^{\pi^a_i} \left[\sum_{q=0}^{\tau-1} r_i(s^{t+q}, a^{t+q}) + V^*_{i,t+\tau}(s^{t+\tau}) \right], \tag{3}$$

where the action sequence is generated by policy i during the time window $[t; t + \tau]$. Similarly, the *losing continuation value* is the least value it achieves upon losing the bidding:

$$\widetilde{V}^{\mathsf{lose}}_{i,t}(s) \coloneqq \min_{j \neq i} \mathbb{E}^{\pi^a_j} \left[\sum_{q=0}^{\tau-1} r_i(s^{t+q}, a^{t+q}) + V^*_{i,t+\tau}(s^{t+\tau}) \right], \tag{4}$$

where the action sequence during $[t; t + \tau]$ is generated by a different policy j using an action policy π^a_j that maximizes j's own winning continuation value. The *control gain* of policy i at (s, t) is

$$G_{i,t}(s) \coloneqq \widetilde{V}^{\mathsf{win}}_{i,t}(s) - \widetilde{V}^{\mathsf{lose}}_{i,t}(s). \tag{5}$$

This quantity measures the advantage of winning control for the next bidding interval rather than losing it. Given bids $b = (b_1, \ldots, b_m)$, let $W(b) \coloneqq \{k \in [1; m] \mid b_k = \max_{\ell \in [1;m]} b_\ell\}$ be the set of highest bidders, and let $p_i(b) = \mathbf{1}[i \in W(b)]/\,|W(b)|$ be the probability that i wins after uniform tie-breaking. For a penalty model $\mathsf{P} \in \{\mathsf{Winner\text{-}Pays}, \mathsf{All\text{-}Pay}\}$, the *one-round bidding payoff* is

$$U^{\mathsf{P}}_{i,t}(s, b) = \begin{cases} \widetilde{V}^{\mathsf{lose}}_{i,t}(s) + p_i(b)\,(G_{i,t}(s) - \rho b_i) & \mathsf{P} = \mathsf{Winner\text{-}Pays}, \\ \widetilde{V}^{\mathsf{lose}}_{i,t}(s) + p_i(b) G_{i,t}(s) - \rho b_i & \mathsf{P} = \mathsf{All\text{-}Pay}. \end{cases} \tag{6}$$

In the Winner-Pays case, the bid cost is paid only in expectation over winning; in the All-Pay case, the bid cost is paid regardless of the winner. Effectively, in the Markov game, the goal of policy i boils down to maximizing the one-round bidding payoff $U^{\mathsf{P}}_{i,t}(s, b)$ at each state s and bidding round t.

A mixed bidding policy for player i at (s, t) is denoted $\pi^b_{i,t}(\cdot \mid s) \in \Delta([0, \beta])$, and a mixed bidding profile is $\pi^b_t(\cdot \mid s) = (\pi^b_{1,t}(\cdot \mid s), \ldots, \pi^b_{m,t}(\cdot \mid s))$. A mixed bidding profile $\pi^{b*}_t(\cdot \mid s)$ is a Nash equilibrium of the one-round bidding game at (s, t) if, for every player i and every alternative bid distribution $\pi^b_{i,t}$, where $\pi^b_{i,t}(\cdot \mid s) \in \Delta([0; \beta])$ for $\mathsf{P} = \mathsf{Winner\text{-}Pays}$ and $\mathsf{P} = \mathsf{All\text{-}Pay}$,

$$\mathbb{E}_{b \sim \pi^{b*}_t(\cdot|s)} \left[U^{\mathsf{P}}_{i,t}(s, b) \right] \geq \mathbb{E}_{b_i \sim \pi^b_{i,t}(\cdot|s),\ b_{-i} \sim \pi^{b*}_{-i,t}(\cdot|s)} \left[U^{\mathsf{P}}_{i,t}(s, b_i, b_{-i}) \right]. \tag{7}$$

For the Winner-Pays setting, the existence of the Nash equilibrium is guaranteed due to the finiteness of the action space $\hat{A}$ [20], and for the All-Pay setting, the same follows from known results in the literature [5]. The equilibrium continuation value is then

$$V^*_{i,t}(s) = \mathbb{E}_{b \sim \pi^{b*}_t(\cdot|s)} \left[U^{\mathsf{P}}_{i,t}(s, b) \right], \tag{8}$$

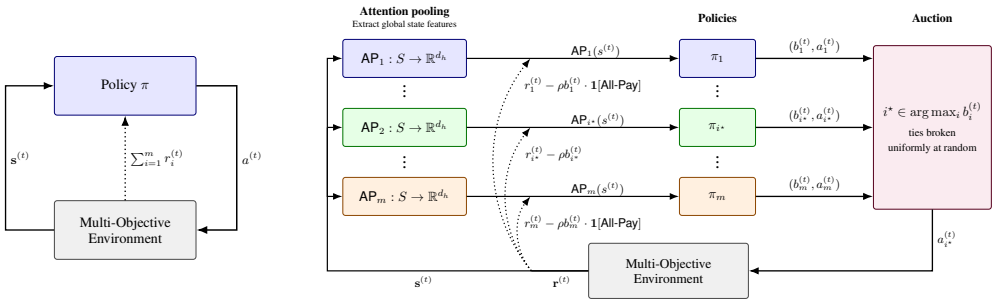
which completes the backward-induction definition.

Theoretical analysis of the bidding schemes. We prove under mild assumptions that if all policies follow the Nash equilibrium then the policy with the highest control gain is guaranteed to obtain control. Consequently, we obtain a greedy solution to maximize the global reward (the sum of local cumulative rewards). Further, from Eq. (4) it follows that the control gains are actually environment-aware, i.e., if the goal of policy i is aligned with other policies (like for the cat feeder, all cats are crowded in one corner), then the control gain of i would be automatically low as it would not mind losing the auction. We state our two results here but defer the proofs to Appendices B and C.

Theorem 1 (Winner-Pays favors high control gain)**.** *In the Winner-Pays setting, consider a single bidding round t at the agent state $s \in S$. Suppose the parameters ρ and β are so chosen that $\rho\beta > \max_{j \in [1;m]} G_{j,t}(s)$. Let i^* be the policy with the largest control gain, such that the following margin condition is fulfilled:*

$$G_{i^*,t}(s) - \max_{j \neq i^*} G_{j,t}(s) > \rho. \tag{9}$$

Then in any pure Nash equilibrium, policy i^* wins control.



(a) Monolithic policy (b) Auction-based multi-policy framework

Figure 2: Comparison between the standard monolithic setting and our auction-based framework.

Theorem 2 (All-Pay favors high control gain)**.** *In the All-Pay setting with $m \geq 3$, consider a single bidding round t at the agent state $s \in S$. Let i^* and j^* be policies for which the following gross gain order holds:*

$$G_{i^*,t} > G_{j^*,t} > \max_{k \notin \{i^*, j^*\}} G_{k,t}. \tag{10}$$

Then there is no Nash equilibrium with pure strategies, although a unique Nash equilibrium exists with mixed strategies. In this Nash equilibrium, policy i^* wins control with probability $1 - {}^{G_{j^*,t}}/_{2G_{i^*,t}}$.

While these results do not automatically imply equivalence to the centralized optimal solution, we obtain an environment-aware, greedy approximation. This provides us a scheduling perspective on the auction: each bidding round picks the objective to receive control based on its urgency. In Appendix A, we prove that in deterministic environments with stationary targets and varying deadlines, a Nash equilibrium in our auction mechanism always selects critically urgent goals, if any. This matches the least slack time first (LSTF) rule, a widely used scheduling heuristic [7].

3 Learning Local Policies in Markov Bidding Games.

Our goal is to learn the local policy of each player in the Markov bidding game from Definition 1 using reinforcement learning. Figure 2b visualizes the architecture of our framework. Yu et al. [32] demonstrate that proximal policy optimization (PPO; [25]) can be effective to train multiple policies concurrently. We follow their recipe and also train the local policies concurrently with one instance of PPO per policy. At each iteration, rollouts are collected from the environment by executing the policies, and each PPO instance uses them. In our experiments, the objectives are symmetric, so we share the actor-critic network across local policies; this reduces the number of learned parameters while preserving separate policy roles through their objective-specific inputs. This sharing also improves sample efficiency: from the same set of rollouts, the shared network receives PPO updates from all m local-policy objectives rather than from a single objective alone. Hyperparameters, network sizes, and rollout settings are reported in Appendix G.

In the practical implementation, we use a discrete bid space rather than the continuous interval in the All-Pay mechanism. Each policy selects its bid from a small finite set of levels, so the bid can be represented as an additional categorical action and trained with the same PPO machinery as the environment action. This discretization is sufficient in our experiments: the bid-level ablation in Figure 5b shows that only a small number of levels is needed for effective coordination.

Additionally, to cope with varying number of objectives we employ the *attention pooling* architecture to compress any number of objectives into a fixed-sized embedding [18, 33]. This design enables the agent to operate with an arbitrary number of policies (i.e., objectives) at deployment time and extract useful information from the global state. Formally, the attention pooling module is a permutation-invariant map $\mathsf{AP} : \bigcup_{m \geq 1} (\mathbb{R}^{d_z})^m \to \mathbb{R}^{d_h}$ from a variable-size collection of objective vectors to a fixed-dimensional embedding. Let $(z_1, \ldots, z_m)$ represent the current state of the system, where z_j denotes the information vector for objective j; for example, z_j may encode the position of a target. Each objective information vector is first processed by a fully connected neural network

f_ψ to produce an embedding $h_j = f_\psi(z_j)$. Given a learned query vector q, we compute attention weights and the pooled embedding as $\alpha_j = \exp(q^\top h_j)/\sum_{\ell=1}^{m} \exp(q^\top h_\ell)$, and $h_{\text{pool}} = \sum_{j=1}^{m} \alpha_j h_j$. The final embedding produced by the module is $\mathsf{AP}(z_1, \ldots, z_m) = h_{\text{pool}}$. Each local policy has its own attention pooling module. The attention-pooling parameters are trained jointly with the rest of the policy network during PPO. This attention mechanism produces a fixed-dimensional representation that summarizes the varying set of objectives.

4 Experimental Results

We evaluate the PPO implementation of our auction framework on evolving reachability objectives and a heterogeneous extension combining feeding priorities and battery charging. We compare against centralized PPO and DWN, and ablate observations, pooling, and auction hyperparameters. A detailed description of the environments and the baseline approaches is provided in Appendix D, below we include a summary. The supplementary material includes our codebase and recorded rollouts of our bidding-based policies on the three base environments.

Environments. We evaluate our approach on three environments: a mobile cat feeder environment, introduced in Section 1, and the two Atari games Assault and Air-raid. In all these environments, the metric being optimized is the cumulative reward across all objectives.

In the Cat Feeder environment, a robot moves on a 30×30 grid to feed up to m cats that move stochastically. Each cat disappears after a fixed number of steps; feeding it yields a reward, while disappearance incurs a penalty of equal magnitude. We train with $m = 8$ cats in the environment and demonstrate generalization up to 14. Each cat is served by a dedicated policy whose observation includes the robot position, the locations of active cats, and their remaining lifetimes. We ablate global observations with attention pooling, global observations with fixed concatenation (no pooling), and local observations only (blind).

In both the Atari games—Assault (Figure 6) and Air-raid—the agent controls a missile launcher to destroy incoming alien ships while avoiding enemy missiles [6]. Although this is not traditionally seen as a multi-objective environment, we treat each ship as a separate objective and assign a local policy to pursue it, enabling a modular and interpretable design. In lieu of the standard pixel observations, we employ the object-centric variant of the Atari environments [10] which provides the environment state information to the policies. Destroying a target yields reward, losing a life incurs a penalty. We do not use attention pooling in the Atari games since there are only up to 3 targets.

Baseline approaches. We compare against two baselines: a single PPO policy optimizing the objective-aggregated reward and deep W-learning (DWN) [23]. These baselines also access the same object-centric observations in the Atari games. The PPO baseline is implemented using CleanRL [14]. Since we provide distance-based reward shaping to help our policies learn, to ensure a fair comparison, we test nearest-target and earliest-expiry distance shaping, as well as urgency-weighted scalarization of objective rewards, in Cat Feeder. Hyperparameters are provided in Appendix G. PPO parameters for Cat Feeder were tuned using 40 Optuna iterations [1] while standard CleanRL Atari hyperparameters were used for the Atari games. The centralized PPO baseline is trained with the same number of environment samples as our method. DWN trains one DQN policy per objective together with W-networks that estimate urgency scores of states. We use the same action window τ for DWN. Training budgets are fixed in advance: 400 PPO iterations for Cat Feeder and Air-Raid, and 150 for Assault. For each of ten runs, one seed initializes the environment and algorithm; evaluation uses subsequent non-overlapping draws from that seeded stream, rather than independently chosen seed identifiers.

Learning dynamics. Figure 3 shows the learning curves for multi-policy bidding and single-policy PPO algorithms on the three environments. While the curves are more noisy with Atari Assault, they exhibit clearly distinguishable sigmoidal shapes with the other two environments. The curves for All-Pay and Winner-Pays methods start growing around the same time. **The variants without attention pooling for targets only start learning much later.** This validates the use of the attention pooling architecture to learn useful embeddings of targets. We remark that, based on the multi-agent PPO recipe of Yu et al. [32], we also found that using a small clip coefficient (~ 0.05) and large batch sizes was crucial to obtain stable but slower convergence of the policies towards an equilibrium.

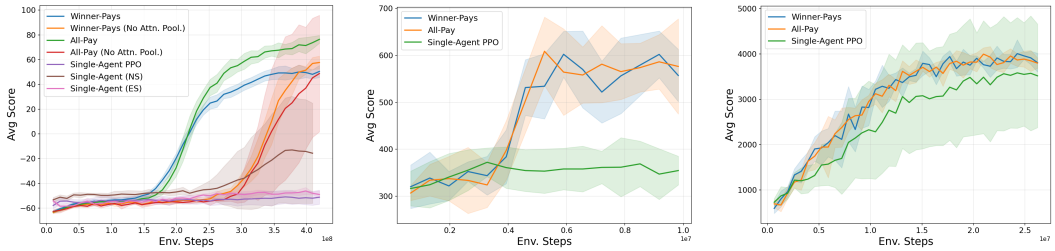


Figure 3: Learning curves for different methods on Cat Feeder (left), Atari Assault (center), and Atari Air-Raid (right). We plot mean over 10 seeds at each evaluation iteration and the shaded region is $\pm$ standard deviation across seeds.

Table 1: Scores: mean (standard deviation) across 10 seeds. For all PPO variants, including local observations only, each seed average uses its last 5 available evaluation checkpoints (evaluations every 10 PPO iterations, 20 rollouts each). Cat Feeder score is completed minus expired requests; Atari scores are game scores.

Method	**Cat Feeder**	**Atari Assault**	**Atari Air-raid**
Winner-Pays	49.44 (4.19)	556.05 (29.23)	**3902.55** (90.44)
w. No Attn. Pool.	51.72 (3.97)	—	—
w. Local Obs. Only	55.80 (11.42)	—	—
All-Pay	**72.52** (4.23)	**576.64** (43.19)	**3870.45** (76.72)
w. No Attn. Pool.	37.72 (50.32)	—	—
w. Local Obs. Only	67.55 (3.79)	—	—
DWN	-10.27 (2.38)	310.80 (159.54)	2438.38 (625.97)
Single-Policy PPO	—	**358.60** (25.99)	**3549.33** (1170.14)
w. No Reward Shaping	-51.59 (5.33)	—	—
w. Nearest Shaping	-14.15 (41.88)	—	—
w. Expiry Shaping	-47.62 (1.52)	—	—
w. Urgency Weighting	51.78 (14.31)	—	—

Comparison with baselines. Table 1 lists performance across the three base environments. Urgency-weighted scalarization raises centralized PPO's Cat Feeder score to 51.78 ± 14.31, comparable in mean to Winner-Pays (49.44 ± 4.19), while All-Pay achieves 72.52 ± 4.23. Thus the stronger baseline changes the comparison materially: All-Pay retains an advantage, whereas Winner-Pays does not outperform urgency-weighted PPO in mean score. Separating objective rewards may ease credit assignment, but these comparisons do not isolate its causal contribution. In Assault, single extracted frames may omit velocities and projectile directions; the four-frame observations in Air-Raid may reduce this partial observability. We did not conduct a dedicated frame-stack ablation, so this remains a possible explanation rather than an established effect.

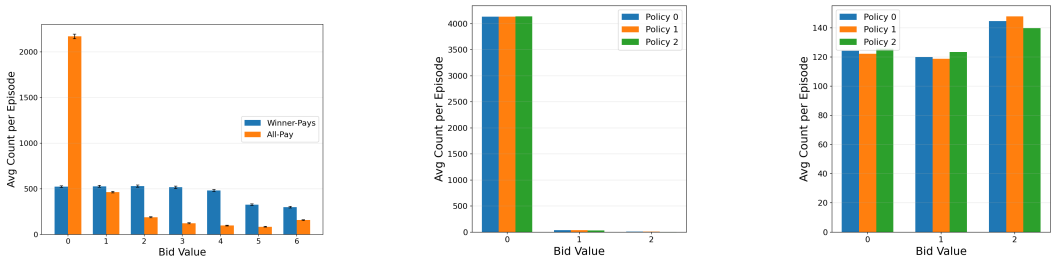
DWN is included as a closely related multi-policy MORL method that learns a state-dependent arbitration signal. Its low scores suggest that TD-error-based arbitration is less effective in these experiments; they do not establish that its architecture alone causes the difference. Our empirical comparisons concern this PPO implementation, rather than a general replacement for PPO across domains.

Observations and pooling. The non-pooled variants retain all target information through fixed concatenation. Winner-Pays scores with and without pooling (49.44 ± 4.19 versus 51.72 ± 3.97) are close relative to the across-seed variation; this does not establish superiority of the non-pooled architecture. Pooling supports variable-size objective sets and earlier learning in the reported curves. All-Pay with pooling has a higher mean than its blind variant (67.55 ± 3.79); blind Winner-Pays has a higher mean (55.80) but greater variability (11.42), so the observations do not show a consistent final-score benefit for both auction mechanisms.

Table 2: Heterogeneous Cat Feeder: mean (standard deviation) of priority-weighted feeding score and battery depletions. Higher feeding scores and fewer depletions are better.

Method	Feeding priority sum	Battery depletions
All-Pay	194.56 (4.67)	0.07 (0.05)
Centralized PPO	111.86 (57.08)	3.81 (5.03)

Heterogeneous objectives and priorities. We extend Cat Feeder to eight cats with feeding priorities from 1 to 4 and three charging stations, all moving stochastically on a 30×30 grid. The robot has a capacity-50 battery; depletion away from a station triggers a tow and refill. One local policy serves each cat, and a separate charging policy receives a battery-dependent effective bid boost (Appendix F.3). Feeding rewards scale with cat priority. Table 2 reports the sum of priorities of fed cats and battery depletions, against centralized PPO with an unweighted reward sum. All-Pay achieves a higher feeding score and fewer battery depletions than centralized PPO, demonstrating coordination across feeding and charging objectives; the nonzero depletion rate does not constitute a safety guarantee. Appendix E.1 reports feeding counts by priority.



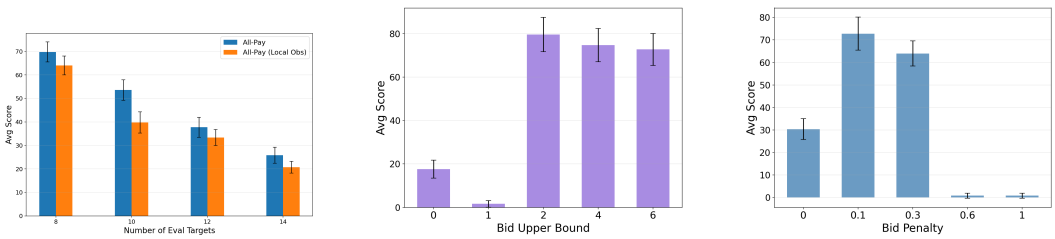
(a) Cat Feeder: Bid distribution

(b) Atari Assault: Bid distribution (Winner-Pays)

(c) Atari Air-Raid: Bid distribution (Winner-Pays)

Figure 4: Comparison of bid distributions across environments and methods. Bars show aggregate bid counts for each bid value across all policies.

Policy behaviors and interpretability. Video rollouts illustrate control allocation as each policy pursues its assigned objective; completion need not occur within one action window. Figure 4 presents the bid distributions. In Assault, bids are usually 0, making allocation close to random tie-breaking; the gains here may arise primarily from objective decomposition rather than learned urgency arbitration. In contrast, the Cat Feeder environment shows a wider bid spread (0–6). Here, All-Pay yields more conservative policies with mostly low bids, while Winner-Pays produces a more even distribution, reflecting greater willingness to bid high. Finally, in Air-raid, we see policies bidding the highest value more often, although they also bid lower values. This is because there is a constant stream of enemy ships that each policy has to destroy and so they want to take control more often to avoid missing out on points.



(a) Scaling number of targets.

(b) Ablation: Bid upper bound

(c) Ablation: Bid penalty

Figure 5: Scaling number of targets and ablations on the Cat Feeder environment.

Scaling number of targets. We conduct an experiment with the Cat Feeder environment in which the number of cats seen during testing is larger than that during training. This is achieved by deploying more copies of the same policy that optimize the additional objectives. Figure 5a evaluates transfer from eight training targets to as many as fourteen testing targets without retraining. The plot uses eight times mean per-episode average performance; scaling selects a single best periodic checkpoint rather than averaging the final five. With fixed episode length, more requests can expire as target count grows (Appendix E.2). The minimum reported per-target completion count remains positive at every tested size (Appendix E.2), supporting continued service across all targets rather than constant performance as their number grows.

Hyperparameter ablations. Figure 5b shows that at least two bid levels are needed for good performance; with only one level, policies cannot compete effectively. Figure 5c evaluates the bid penalty factor: without penalties all policies overbid, hurting performance, while overly large penalties discourage bidding. Additional action-window ablations are reported in Appendix E.

5 Concluding Remarks

We propose a novel modular framework for solving multi-objective RL tasks, where each objective is pursued by a dedicated local policy, and policies bid for the right to execute their action. The advantages of the framework over the baselines include: better learning and policy performances, and superior adaptability in the face of appearing and disappearing objectives at runtime.

Limitations. Our auction-based mechanism selects policies based on the relative urgency of the most urgent policy compared to the rest. In doing so, sometimes the framework could sacrifice overall optimality: in the cat feeder example, if the most urgent target is isolated in one corner, and the rest are concentrated in the opposite corner, the agent could prioritize the single urgent task over the *majority* of less urgent tasks, even if pursuing the majority would lead to greater overall payoff.

Future directions. Several directions exist. First, we will attempt to mitigate the above limitation by extending our auction-mechanism to include "group urgencies" into considerations, e.g., by incorporating mechanisms like majority voting. Second, we assumed that all objectives belong to the same family, which we will relax in the future. In some cases, a lexicographic ordering of priorities over objective types would be interesting, like safety usually gets higher priority than the rest. Finally, in our experiments, we observe that the auction mechanism treats all objectives *fairly*, whereas monolithic policies tend to focus on one or few objectives. If true in general, it could imply that our framework achieves much more than merely maximizing the sum of individual payoffs: we will continue investigating this in future work.

Acknowledgments and Disclosure of Funding

Guruprerana Shabadi is supported by NSF Award SLES 2331783. Kaushik Mallik is supported by the grant RYC2024-049116-I funded by MICIU/AEI/10.13039/501100011033 and the ESF+.

References

[1] 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.

[2] Guy Avni, Thomas A Henzinger, and Ventsislav Chonev. Infinite-duration bidding games. *Journal of the ACM (JACM)*, 66(4):1–29, 2019.

[3] Guy Avni, Kaushik Mallik, and Suman Sadhukhan. Auction-based scheduling. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 153–172. Springer, 2024.

[4] Guy Avni, Martin Kurečka, Kaushik Mallik, Petr Novotnỳ, and Suman Sadhukhan. Bidding games on markov decision processes with quantitative reachability objectives. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pages 161–169, 2025.

[5] Michael R Baye, Dan Kovenock, and Casper G De Vries. The all-pay auction with complete information. *Economic Theory*, 8(2):291–305, 1996.

[6] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

[7] Kyle Brown, Oriana Peltzer, Martin A Sehr, Mac Schwager, and Mykel J Kochenderfer. Optimal sequential task assignment and path finding for multi-agent robotic assembly planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 441–447. IEEE, 2020.

[8] Woohyeon Byeon, Giseung Park, Jongseong Chae, Amir Leshem, and Youngchul Sung. Multi-objective reinforcement learning with max-min criterion: A game-theoretic approach. *arXiv preprint arXiv:2510.20235*, 2025.

[9] Yun Chen, Jinfeng Wu, Chaoshuai He, and Si Zhang. Intelligent warehouse robot path planning based on improved ant colony algorithm. *IEEE Access*, 11:12360–12367, 2023.

[10] Quentin Delfosse, Jannis Blüml, Bjarne Gregori, Sebastian Sztwiertnia, and Kristian Kersting. Ocatari: Object-centric atari 2600 reinforcement learning environments, 2023.

[11] Marcin Dziubiński and Krzysztof Jahn. Discrete two player all-pay auction with complete information. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 2659–2666, 2023.

[12] Maria Fox, Alfonso Gerevini, Derek Long, Ivan Serina, et al. Plan stability: Replanning versus plan repair. In *ICAPs*, volume 6, pages 212–221, 2006.

[13] Saul Gass and Thomas Saaty. The computational algorithm for the parametric objective function. *Naval research logistics quarterly*, 2(1-2):39–45, 1955.

[14] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

[15] Mark Humphrys. W-learning: Competition among selfish q-learners. 1995.

[16] Martin Klissarov, Akhil Bagaria, Ziyan Luo, George Konidaris, Doina Precup, and Marlos C Machado. Discovering temporal structure: An overview of hierarchical reinforcement learning. *arXiv preprint arXiv:2506.14045*, 2025.

[17] Andrew J Lazarus, Daniel E Loeb, James G Propp, Walter R Stromquist, and Daniel H Ullman. Combinatorial games under auction play. *Games and Economic Behavior*, 27(2):229–264, 1999.

[18] 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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR, 2019.

[19] Beatriz M Méndez-Hernández, Erick D Rodríguez-Bazan, Yailen Martinez-Jimenez, Pieter Libin, and Ann Nowé. A multi-objective reinforcement learning algorithm for jssp. In *International Conference on Artificial Neural Networks*, pages 567–584. Springer, 2019.

[20] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[21] Giseung Park, Woohyeon Byeon, Seongmin Kim, Elad Havakuk, Amir Leshem, and Youngchul Sung. The max-min formulation of multi-objective reinforcement learning: From theory to a model-free algorithm. *arXiv preprint arXiv:2406.07826*, 2024.

[22] Matteo Pirotta, Simone Parisi, and Marcello Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[23] Juan C Rosero, Nicolás Cardozo, and Ivana Dusparic. Multi-objective deep reinforcement learning optimisation in autonomous systems. In *2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, pages 97–102. IEEE, 2024.

[24] Stuart J Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 656–663, 2003.

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

[26] Yi Shen, Benjamin McClosky, Joseph W Durham, and Michael M Zavlanos. Multi-agent reinforcement learning for resource allocation in large-scale robotic warehouse sortation centers. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 7137–7143. IEEE, 2023.

[27] Umer Siddique, Paul Weng, and Matthieu Zimmer. Learning fair policies in multi-objective (deep) reinforcement learning with average and discounted rewards. In *International Conference on Machine Learning*, pages 8905–8915. PMLR, 2020.

[28] Leo K Simon and William R Zame. Discontinuous games and endogenous sharing rules. *Econometrica*, 58(4):861–872, 1990. doi: 10.2307/2938353.

[29] Ramanjeet Singh, Jing Ren, and Xianke Lin. A review of deep reinforcement learning algorithms for mobile robot path planning. *Vehicles*, 5(4):1423–1451, 2023.

[30] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[31] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 191–199. IEEE, 2013.

[32] 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.

[33] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, pages 3391–3401, 2017.

A A Special Case of MO-MDPs: Deterministic Time-Bounded Reachability with Multiple Stationary Goals

We now isolate a deterministic path-planning special case that provides useful intuition for the auction mechanism. This setting is simple enough that we can even show that our auction-based framework would coincide with one of the standard algorithms (called LSTF, introduced later) for solving this problem (see Sec. A.2). This shows that, while in the most general MO-MDP setting we are not able to make theoretical claims on optimality, additional assumptions sometime will bridge this gap.

A.1 The Model and the Shape of the Optimal Policy

The special case can be stated directly as follows. An agent moves without uncertainty in a location space L, starting from $l_0 \in L$. There are m stationary goals $g_1, \dots, g_m$, where goal g_i has deadline $d_i \in \mathbb{N}_{>0}$, reward $r_i \in \mathbb{R}_{>0}$, and missed-deadline penalty $-M$. Let $d(x, g_i)$ denote the minimum travel time from location x to g_i. The agent receives r_i if it reaches g_i before d_i, and receives $-M$ otherwise.

Equivalently, the state is a pair $(x, t) \in L \times \mathbb{N}_{\geq 0}$, where x is the agent location and t is time. Actions are movement commands in L, the transition map is deterministic, the initial state is $(l_0, 0)$, and the finite horizon H is chosen large enough to include the relevant deadlines. The objective is to maximize the sum of the local rewards over all goals.

Proposition 1. *In the deterministic stationary-goal setting, there exists a permutation $\sigma^* \in \Sigma_m$, where Σ_m is the set of all permutations of $[1; m]$, such that an optimal policy pursues the goals in the order*

$$\sigma^*(1), \dots, \sigma^*(m),$$

and follows shortest paths from l_0 to $g_{\sigma^*(1)}$ and from $g_{\sigma^*(q)}$ to $g_{\sigma^*(q+1)}$ for every $q \in [1; m-1]$.

Proof. Fix any deterministic policy and consider the order in which it first reaches goals along one of its induced trajectories. Since goals are stationary and rewards depend only on first arrival before the corresponding deadlines, any subpath between two consecutive first-reached goals can be replaced by a shortest path between the same endpoints. This replacement cannot delay any reached goal and therefore cannot decrease the total reward. Repeating this shortcutting argument yields a policy that visits goals according to some permutation and uses shortest paths between consecutive goals. Applying the argument to an optimal policy gives the claimed permutation σ^*. □

Thus, in this deterministic special case, policy synthesis reduces to choosing an ordering of the goals. Computing the globally optimal ordering is a centralized scheduling problem, and it can be expensive even in deterministic metric settings. This motivates lightweight priority rules that can be implemented by local policies.

A standard deterministic priority rule is *least slack time first* (LSTF) [7], which schedules the goal with smallest nonnegative slack

$$slack_i(x, t) \coloneqq d_i - t - d(x, g_i).$$

Slack is well matched to stationary goals and deterministic travel times, but it does not directly apply to the full MO-MDP setting with stochastic motion, discounted rewards, missed-deadline penalties, and objectives that may appear or disappear. For the general setting, our theory instead uses the control gain of each local policy: the marginal continuation-value advantage of granting that policy the next control interval rather than deferring control to another policy. This reward-aware urgency captures both deadline pressure and whether another policy's action would incidentally help the same objective.

A.2 The Guarantees Provided by the Auction-Based Framework

We now connect the auction mechanism to least slack time first (LSTF) in the deterministic stationary-goal setting from the previous section. Recall that each goal g_i is stationary, the agent moves deterministically, and $d(x, g_i)$ is the minimum travel time from location x to goal g_i. At location x and time t, define the slack of goal i by

$$slack_i(x, t) \coloneqq d_i - t - d(x, g_i).$$

Once g_i is reached, it disappears from the map and we write $slack_i(x,t) = \perp$. The least slack time first rule schedules a goal with minimum nonnegative slack [7]. Formally, at location x and time t, LSTF selects any goal

$$i_{\mathrm{LSTF}} \in \arg\min_{i:slack_i(x,t)\geq 0} slack_i(x,t),$$

with ties broken arbitrarily. Thus LSTF gives control to the goal whose deadline is closest after accounting for travel time.

We show that, when one goal is critically urgent and all other goals are safely non-urgent, the auction selects the same goal as LSTF. At each bidding round, every goal i is represented by a local policy that proposes an action and submits a nonnegative bid. The highest bidder controls the agent for the next $\tau \in \mathbb{N}_{>0}$ time steps, with ties broken uniformly at random. Bids are bounded by β, and $\rho > 0$ is the bid penalty coefficient. For Winner-Pays, bids are integer values in $[0;\beta]$; for All-Pay, bids are real values in $[0,\beta]$. We analyze two payment rules: in Winner-Pays, only the winning policy pays ρb_i, while in All-Pay, every policy pays ρb_i.

In the remainder of this appendix, $s \in L$ denotes the current agent location and t denotes the current time. Suppose π_i is the local policy for the i-th goal, mapping each pair (s,t) to a bid b_i and an action a_i. We write $\pi_i^b\colon (s,t) \mapsto b_i$ and $\pi_i^a\colon (s,t) \mapsto a_i$ for the bidding and action components. In this deterministic setting, π_i^a proceeds toward its goal g_i along a shortest path, and $T^\tau(s,\pi_i^a)$ denotes the state reached by applying π_i^a for τ steps from s.

For a bidding round at time t, define $V_i^{\mathsf{win}}(s,t,b_i)$ and $V_i^{\mathsf{lose}}(s,t,b_i)$ as the value to policy i when it bids b_i and respectively wins or loses the next bidding interval. Let $V_i^{\mathsf{cont}}(s,t)$ denote the Nash equilibrium continuation value of the subgame starting from (s,t). The value functions are defined by backward induction as

$$\begin{aligned} V_i^{\mathsf{win}}(s,t,b_i) &\coloneqq \gamma^\tau V_i^{\mathsf{cont}}(T^\tau(s,\pi_i^a), t+\tau) - \rho b_i, \\ V_i^{\mathsf{lose}}(s,t,b_i) &\coloneqq \gamma^\tau V_i^{\mathsf{cont}}(T^\tau(s,\pi_j^a), t+\tau) - \rho b_i \cdot \mathbf{1}[\mathsf{All\text{-}Pay}], \end{aligned}$$

where j is the policy that wins control when i loses. The net gain from winning rather than losing is

$$\begin{aligned} \Delta_i^{\mathrm{net}}(s,t,b_i) &\coloneqq V_i^{\mathsf{win}}(s,t,b_i) - V_i^{\mathsf{lose}}(s,t,b_i) \\ &= \Delta_i^{\mathrm{gross}}(s,t) - \rho b_i \cdot \mathbf{1}[\mathsf{Winner\text{-}Pays}], \end{aligned}$$

where the gross gain

$$\Delta_i^{\mathrm{gross}}(s,t) \coloneqq \gamma^\tau \left[V_i^{\mathsf{cont}}(T^\tau(s,\pi_i^a), t+\tau) - V_i^{\mathsf{cont}}(T^\tau(s,\pi_j^a), t+\tau)\right]$$

measures the value advantage to objective i of winning the next bidding interval.

Lemma 1. *In both Winner-Pays and All-Pay settings, if policy i^* is in the critical regime, i.e., $slack_{i^*}(s,t) \in [0,\tau]$, and every other policy $j \neq i^*$ is in the safe regime, i.e., $slack_j(s,t) > \tau$, then*

$$\Delta_{i^*}^{\mathrm{gross}}(s,t) - \max_{j\neq i^*} \Delta_j^{\mathrm{gross}}(s,t) > M - r_{\max} > 0. \tag{11}$$

Proof. Since $slack_{i^*}(s,t) \leq \tau$, losing the bidding interval causes i^* to miss its deadline unless the winning policy reaches g_{i^*} incidentally during that interval. In the worst case represented by the losing continuation value, this incidental help does not occur. Hence $V_{i^*}^{\mathsf{lose}}(s,t,b_{i^*}) = -M - \rho b_{i^*} \cdot \mathbf{1}[\mathsf{All\text{-}Pay}]$. On the other hand, if i^* wins the bidding, then $V_{i^*}^{\mathsf{win}}(s,t,b_{i^*}) \geq -\rho b_{i^*}$, with strict inequality if the goal is reached during the next τ steps. Therefore

$$\Delta_{i^*}^{\mathrm{gross}}(s,t) \geq M.$$

For any $j \neq i^*$, the safe-regime condition $slack_j(s,t) > \tau$ implies that losing the next bidding interval does not by itself force j to miss its deadline. Thus $V_j^{\mathsf{win}}(s,t,b_j) \leq \gamma^{\tau+1} r_j - \rho b_j$ and $V_j^{\mathsf{lose}}(s,t,b_j) \geq -\rho b_j \cdot \mathbf{1}[\mathsf{All\text{-}Pay}]$, so

$$\Delta_j^{\mathrm{gross}}(s,t) \leq \gamma^{\tau+1} r_j < r_j \leq r_{\max}.$$

Combining the two bounds gives

$$\Delta_{i^*}^{\mathrm{gross}}(s,t) - \max_{j\neq i^*} \Delta_j^{\mathrm{gross}}(s,t) > M - r_{\max}.$$

Since $M > r_{\max}$, the right-hand side is positive. □

Lemma 1 is the only special-case ingredient needed to relate the auction mechanism to LSTF. It says that a critically urgent goal has a strictly larger gross gain than every safe goal. Therefore, in the Winner-Pays setting, if $0 < \rho < M - r_{\max}$, the margin condition in Theorem 1 is satisfied and the critically urgent LSTF goal wins in any pure Nash equilibrium. In the All-Pay setting, the same gross-gain ordering is the input to Theorem 2; hence the mixed equilibrium concentrates control on the critically urgent LSTF goal, with concentration increasing as the urgent goal's gross gain dominates the runner-up's gross gain.

B Proof of the Winner-Pays Control-Gain Result

Proof of Theorem 1. Suppose for contradiction, some other policy $j \neq i^*$ wins control by bidding b_j, while policy i^* bids $b_{i^*} < b_j$. As this is a Nash equilibrium, it must hold that $U^{\mathsf{P}}_{j,t}(s, b_j) > \widetilde{V}^{\mathsf{lose}}_{j,t}(s)$, i.e., $G_{j,t}(s) - \rho b_j > 0$, because otherwise, policy j would be better off by losing the bidding instead. This gives us the upper bound $\rho b_j < G_{j,t}(s)$. It follows from $\rho\beta > \max_{k\in[1;m]} G_{k,t}(s)$ that $\rho b_j < \rho\beta$, and hence $b_j < \beta$, i.e., the policy j does not make the maximum bid.

Now we assumed that policy i^* had bid $b_{i^*} < b_j$, but we show that policy i^* would achieve a higher one-round bidding payoff by bidding $b_j + 1$ (valid, since $b_j < \beta \implies b_j + 1 \leq \beta$), contradicting the Nash equilibrium: $U^{\mathsf{P}}_{i^*,t}(s, b_j+1) = \widetilde{V}^{\mathsf{lose}}_{i^*,t}(s)+G_{i^*,t}(s)-\rho(b_j+1) = \widetilde{V}^{\mathsf{lose}}_{i^*,t}(s)+G_{i^*,t}(s)-\rho b_j-\rho > \widetilde{V}^{\mathsf{lose}}_{i^*,t}(s) + G_{i^*,t}(s) - G_{j,t}(s) - \rho > \widetilde{V}^{\mathsf{lose}}_{i^*,t}(s) = U^{\mathsf{P}}_{i^*,t}(s, b_j)$. □

C Proof of the All-Pay Control-Gain Result

Proof of Theorem 2. It follows from the work of Baye et al. [5] that in the Nash equilibrium, the optimal bid of the policies are as follows. The policy i^* picks a uniformly random bid b^*_i from $[0, G_{j^*,t}/\rho]$, the policy j^* picks a uniformly random bid b^*_j from $(0, G_{j^*,t}/\rho]$ while picking the bid 0 with probability $(G_{i^*,t} - G_{j^*,t})/G_{i^*,t}$, and every other policy $k \notin \{i^*, j^*\}$ deterministically picks the bid 0. Clearly, if $b^*_i > b^*_j$, then i^* wins control, and we need to show that $P[b^*_i > b^*_j] = 1 - {}^{G_{j^*,t}}/_{2G_{i^*,t}}$. To show this, consider the probability density function $f_{i^*}(b) = {}^{\rho}/_{G_{j^*,t}}$ of b_{i^*}, and the cumulative distribution function $F_{j^*}(b) = (G_{i^*,t} - G_{j^*,t})/G_{i^*,t} + {}^{\rho b}/_{G_{i^*,t}}$ of b_{j^*}. Then,

$$
\begin{aligned}
P[b^*_i > b^*_j] &= \int_{-\infty}^{\infty} F_{j^*}(b) f_{i^*}(b) db \\
&= \int_0^{G_{j^*,t}/\rho} \left(\frac{G_{i^*,t} - G_{j^*,t}}{G_{i^*,t}} + \frac{\rho b}{G_{i^*,t}} \right) \cdot \frac{\rho}{G_{j^*,t}} \cdot db \\
&= \int_0^{G_{j^*,t}/\rho} \frac{G_{i^*,t} - G_{j^*,t}}{G_{i^*,t}} \cdot \frac{\rho}{G_{j^*,t}} \cdot db + \int_0^{G_{j^*,t}/\rho} \frac{\rho b}{G_{i^*,t}} \frac{\rho}{G_{j^*,t}} \cdot db \\
&= \frac{G_{i^*,t} - G_{j^*,t}}{G_{i^*,t}} \cdot \frac{\rho}{G_{j^*,t}} \cdot \frac{G_{j^*,t}}{\rho} + \frac{\rho^2}{G_{i^*,t} G_{j^*,t}} \cdot \frac{(G_{j^*,t})^2}{2\rho^2} \\
&= \frac{G_{i^*,t} - G_{j^*,t}}{G_{i^*,t}} + \frac{G_{j^*,t}}{2G_{i^*,t}} \\
&= 1 - \frac{G_{j^*,t}}{G_{i^*,t}} + \frac{G_{j^*,t}}{2G_{i^*,t}} \\
&= 1 - \frac{G_{j^*,t}}{2G_{i^*,t}}.
\end{aligned}
$$

□

D Experimental Setup (Detailed)

We consider three multi-objective environments for our empirical studies, namely a mobile cat feeder and the Atari games Assault and Air-Raid. These environments are described below.

Environment I: Cat Feeder. The environment is as described in Section 1 and is modeled using a 30×30 grid with up to m cats that are moving continuously according to unknown stochastic



Figure 6: Atari Assault environment.

processes. The value of m was set as 8 during the training time, whereas it was varied up to 14 during the testing time. Each cat disappears after a given number of time steps. If the agent reaches a cat before it disappears, it receives a reward, and otherwise it receives a penalty of equal magnitude. To avoid sparsity of the reward function, we provide each policy a small reward for going closer to its target. Performance is measured as the difference between the number of feeding requests completed and the number of those that expired. The Cat Feeder environment parameters are listed in Table 5, and the corresponding multi-policy, single-policy PPO, and DWN training parameters are listed in Tables 6 to 8.

In our modular framework, each cat will be chased by a dedicated local policy. The observation space of each local policy includes the current position of the robot, the current locations of the active requests, and the time steps until the expiration of the requests. For local policy i, the Cat Feeder reward is

$$r_i = r_i^{\text{bid}} + r_i^{\text{window}} + r_i^{\text{dist}} + r_i^{\text{reach}} + r_i^{\text{expire}}.$$

Here r_i^{bid} is the auction payment term, r_i^{window} is the action-window penalty, r_i^{dist} rewards distance improvement toward policy i's target, r_i^{reach} is paid when that target is reached, and r_i^{expire} penalizes target expiration. The multi-policy decomposition preserves a separate reward signal for each target. Centralized PPO can also scalarize per-target rewards, but must infer control allocation and movement from the resulting scalar signal. The single-policy PPO baseline uses the distance-improvement, target-reached, and target-expiration terms, without bid or action-window penalties.

Environment II: Atari Assault. We consider the Atari game called Assault, where the agent controls a missile launcher to shoot oncoming alien ships while avoiding enemy missiles. The standard approach is to view this as a single-objective environment, where the agent's policy observes the entire image or the 128 byte raw RAM state observations to determine the optimal actions [6].

For our modular framework, we break the objective down to a multi-objective problem, where each alien ship is viewed as a separate objective, and each local policy attempts to destroy its assigned target while avoiding enemy missiles. Not only does this facilitate an improvement in the performance, as we will demonstrate, but also we obtain a transparent design, because which target is being chased is revealed at each point. This modular design is made possible by the object-centric variant of the Atari environment [10], which exposes the states of each individual target and the missiles by decoding the RAM states. The full observation space of each local policy also include the current position of the missile launcher, the current health, and the number of remaining lives. The local policy gets a reward or a penalty upon destroying its target enemy ship or losing a life, respectively. For local policy i, the Assault reward is

$$r_i = r_i^{\text{destroy}} + r_i^{\text{life}} + r_i^{\text{overheat}} + r_i^{\text{hot}} + r_i^{\text{bid}} + r_i^{\text{window}}.$$

The destroy reward is credited to the policy assigned to the destroyed enemy, while life-loss, overheat, and fire-while-hot penalties discourage unsafe firing behavior. The bid and action-window terms account for the auction mechanism. Lastly, we use the game score as the metric for evaluation. The Assault environment parameters are listed in Table 9, and the corresponding multi-policy, single-policy PPO, and DWN training parameters are listed in Tables 10 to 12.

Environment III: Atari Air-Raid. Air-Raid uses the same modular Atari setup: each incoming enemy is treated as an objective and assigned to a local policy. Positive score changes are interpreted as enemy kills, and each kill yields a destroy reward proportional to the score increase. We also penalize near misses from enemy missiles: if an enemy missile enters a fixed box around the player without causing a life loss, the controlling policy receives a near-hit penalty. For local policy i, the Air-Raid reward is

$$r_i = r_i^{\text{destroy}} + r_i^{\text{near-hit}} + r_i^{\text{bid}}.$$

Destroy rewards are credited to the policy associated with the hit enemy when that policy owns the player missile that caused the score event, near-hit penalties are assigned to the policy currently controlling the player, and bid penalties are applied according to the auction rule. The single-policy PPO baseline uses the destroy and near-hit terms without bid penalties. Additional training and implementation details are reported in Appendix G.

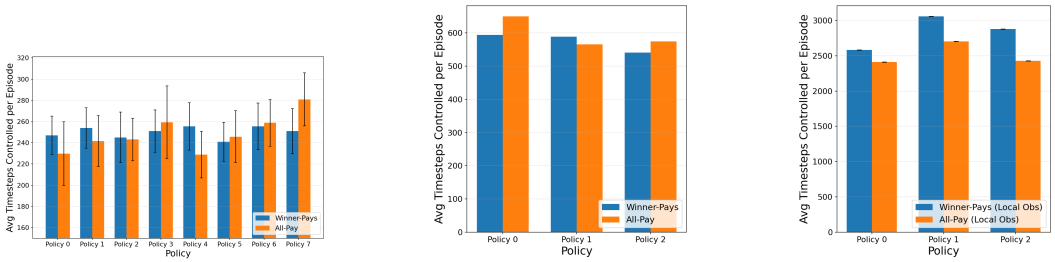
Baseline approaches. We compare our method against two baselines: a single policy PPO with weighted rewards and deep W-learning (DWN; [23]). For the original PPO baselines in the homogeneous environments, we use the total reward across objectives at each step. We additionally evaluate urgency-weighted scalarization in Cat Feeder, where objective reward components are weighted dynamically by urgency. The precise weighting and update order are given in Appendix F.2. We note that in Atari Assault and Air-raid, the single-policy PPO reward includes enemy-destroy rewards, raw score rewards, life-loss penalties, overheat penalties, and fire-while-hot penalties, but no bid or action-window terms. The observation space remains the same as the multi-policy setting. In the Cat Feeder environment, we implement three reward mechanisms to help guide learning: one only with the sparse rewards from the environment, one with distance-based dense reward with respect to the nearest feeding request (denoted Nearest Shaping/NS in the experiments), and finally one where we provide distance-based reward with respect to the request that has the least amount of time until expiration (denoted Expiry Shaping/ES). The single-policy PPO baseline does not use an action window: because there is no policy-switching decision, it selects a new action at every environment step. The action window τ applies only to multi-policy methods, where it determines how long the selected local policy retains control after a bidding or W-network selection event. The PPO implementation is taken from CleanRL [14]. We use the specified PPO hyperparameters from CleanRL for the Atari games and we tune the hyperparameters for the Cat Feeder environment with 40 Optuna iterations [1].

DWN is another multi-policy method for multi-objective RL that uses DQN for each of the local policies and additionally learns a W-network for each policy that predict scalar weights. These weights are used to choose which policy's action should be played at each step. For temporal consistency among multi-policy methods, DWN uses the same action window τ as our bidding methods. The W-network is trained alongside the Q-network by updating it with a gradient step to predict the negative temporal difference (TD) error when the policy does not get to play its action. The assertion here is that the negative TD-error quantifies how important it is for the policy to play its action. The implementation is borrowed from the repository of [23]. All hyperparameters for the baselines are listed in Appendix G.

Heterogeneous Cat Feeder. The extension adds priority-scaled feeding rewards and a distinct charging objective to the same gridworld. Eight cat policies compete with one charging policy; the three charging stations and the cats move stochastically. The charging policy has a battery-dependent effective bid boost to encode its known priority. Battery dynamics, effective bids, network sharing, and the centralized reward are specified in Appendix F.3. The feeding metric sums priorities of completed requests (priorities 1–4), and battery depletions are reported separately.

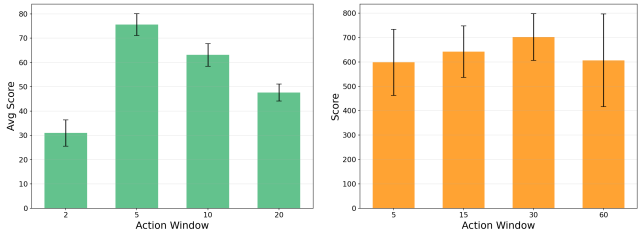
E Additional Experimental Results

Figure 8 studies the action-window size in the Cat Feeder and Atari Assault environments. Smaller windows cause excessive switching between policies, while larger windows prevent sufficient switching to serve all objectives. The best performance occurs at an intermediate action-window size.



(a) Cat Feeder. (b) Atari Assault. (c) Atari Air-Raid.

Figure 7: Control distributions across environments and methods. The control distribution is the number of timesteps controlled by each policy.



(a) Cat Feeder. (b) Atari Assault.

Figure 8: Effect of the action-window size on performance.

E.1 Heterogeneous Cat Feeder: Feeding Counts by Priority

Table 3 breaks down completed feeding requests by priority in the heterogeneous environment. The priority-weighted score in Table 2 is the sum of these completion counts weighted by their priorities. The results show service across all four priority levels while the charging policy competes for control.

Table 3: Completed feeding requests by priority: mean (standard deviation). Priority 4 is the highest feeding priority.

Method	Priority 1	Priority 2	Priority 3	Priority 4
All-Pay	17.45 (0.32)	18.75 (0.70)	19.56 (0.73)	20.23 (0.64)
Centralized PPO	6.82 (1.55)	8.61 (3.12)	11.72 (6.13)	13.17 (7.77)

E.2 Target Coverage under Scaling

Table 4 reports the minimum per-target completion count when increasing the number of targets at deployment, averaged over twenty evaluation episodes. For episode e and target i, let $C_{e,i}$ be its completion count. The reported statistic is

$$\overline{C}_{\min} = \frac{1}{E} \sum_{e=1}^{E} \min_{i} C_{e,i}, \qquad E = 20. \tag{12}$$

The minimum is taken within each episode before averaging, rather than taking the minimum of per-target episode averages. These values are read from the saved JSON field `statistics.avg_min_targets_reached` and recompute from the twenty episode-level minima. Scaling selects the single periodic checkpoint with the highest `avg_avg_performance`, as detailed in Appendix F.4. The model is trained with eight targets and transferred by adding copies of the learned policy, each conditioned on its assigned target. The episode length remains fixed, so additional targets divide the available service time and may expire before being reached. Positive coverage at every tested size indicates that no target is entirely excluded in this aggregate, but does not imply equal service or invariant performance.

Table 4: Scaling Cat Feeder: mean of the per-episode minimum target-completion counts across twenty episodes.

Number of targets	8	10	12	14
Minimum completion count	8.2	5.7	5.9	4.1

F Additional Implementation Details

F.1 Parameter Sharing

In our homogeneous-objective experiments, the objectives are symmetric: each local policy solves the same type of reachability task, but for a different assigned target. We therefore share the actor-critic network parameters across local policies. This reduces the number of learned parameters and lets the shared network receive PPO updates from all local-policy objectives.

Parameter sharing does not make the local policies indistinguishable. For each local policy i, the policy input contains both a pooled embedding of the full set of objective vectors and the objective-specific vector z_i for that policy's assigned target. Thus two policy copies share network weights, but are conditioned on different z_i's and receive different local rewards. In Cat Feeder, z_i encodes the assigned feeding request, including its position and remaining lifetime; in the Atari environments, z_i encodes the assigned enemy object exposed by the object-centric observation.

F.2 Urgency-Weighted Scalarization

Let L denote the target expiry interval (`target_expiry_steps`), distinct from the episode horizon H, and let $c_{i,t}$ be target i's age counter in the state before action t. Define its clamped remaining lifetime, urgency, and normalized weight by

$$\ell_{i,t} = \max\{L - c_{i,t}, 1\}, \qquad q_{i,t} = \frac{\mathbf{1}[i \text{ active at } t]}{\ell_{i,t}}, \tag{13}$$

$$w_{i,t} = \begin{cases} q_{i,t} / \sum_j q_{j,t}, & \sum_j q_{j,t} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

The weights are computed before movement, counter increments, expiration, or respawn. Active weights sum to one, reached static targets receive zero weight, and there is no additional normalization by the number of targets. For example, two active targets with remaining lifetimes 1 and 4 receive weights 0.8 and 0.2. The scalar reward is

$$r_t = \sum_i w_{i,t} r_{i,t}^{\text{obj}} - \lambda_B \mathbf{1}[\text{battery depletion at } t], \tag{15}$$

where $r_{i,t}^{\text{obj}}$ includes distance-improvement shaping, priority-weighted completion reward, and the expiry penalty. The depletion term applies only in battery-enabled environments and is not multiplied by target urgency weights.

F.3 Heterogeneous Cat Feeder Implementation

Battery and stations. The reported moving-station comparison uses battery capacity 50. The three stations start at $(0, 0)$, $(15, 15)$, and $(29, 29)$ and follow independent bounded persistent random walks, moving every 15 steps with direction-change probability 0.1. A successful one-cell movement consumes one battery unit; boundary or no-movement steps consume none. Occupying a charging station immediately restores the battery to capacity. Reaching zero away from a station causes an immediate tow to the nearest station and a full refill; we count such events as battery depletions. A capacity-40 fixed-station reference configuration is a different benchmark and is not the configuration used for the reported moving-station results.

Effective bids and payments. Feeders and charger both choose raw categorical bids in $\{0, \ldots, 6\}$. Let B_t be the current battery level and e_t the energy required to reach the nearest station. The charger

is forced to bid zero unless $B_t \le e_t + 8$. For a permitted raw charger bid $b_{\mathrm{chg},t}$, the bid used to determine the winner is

$$b^{\mathrm{eff}}_{\mathrm{chg},t} = b_{\mathrm{chg},t} + 2\mathbf{1}[b_{\mathrm{chg},t} > 0 \ \wedge \ B_t \le 25]. \tag{16}$$

Thus the effective charger ceiling is 8 when the boost applies, versus 6 for feeders. The All-Pay payment uses the raw bid, not the boosted effective bid. This battery-dependent priority rule is an implementation extension to the common-cap auction analyzed in the theory.

Parameter sharing. The eight feeder controllers share one actor–critic network, with observations reordered according to the assigned target identity. The charger uses a completely separate actor–critic with separate direction and bid actor trunks in the moving-station experiment. No parameters are shared between feeders and charger.

Centralized baseline. The unassisted centralized PPO baseline in Table 2 directly chooses body movement without bidding, an action window, or an execution-time action override. Let $d_{i,t}$ be distance to target i, $p_i \in \{1, 2, 3, 4\}$ its feeding priority, and D_t the battery-depletion indicator. Its unweighted scalar reward is

$$\begin{aligned} r_t = \sum_i \big[& 0.6(d_{i,t} - d_{i,t+1}) + 20 p_i \mathbf{1}[i \text{ reached at } t] \\ & - 50\mathbf{1}[i \text{ expired at } t] \big] - 10 D_t. \end{aligned} \tag{17}$$

An assisted centralized variant adds reserve-gated progress shaping toward a station, a refill bonus proportional to the restored battery fraction, and an additional depletion penalty of 100. It also has no execution-time action override; these auxiliary terms are not part of the unassisted baseline reported here.

F.4 Evaluation Aggregation

For Table 1, PPO results first average each seed's last five available evaluation checkpoints, then report the mean and sample standard deviation across seed averages. The local-observation ablation (`visible_targets=0`, called blind in the discussion) uses the same aggregation as the other PPO variants. Scaling evaluations use a different checkpoint-selection procedure: a single periodic checkpoint is selected by the largest `avg_avg_performance`, rather than averaging the final five checkpoints. This selection uses evaluation performance and should be distinguished from evaluating a prespecified final checkpoint on an independent test set. The scaling plot displays eight times the mean per-episode average performance, which is distinct from the minimum-completion statistic defined in Appendix E.2.

G Reproducibility Details

The code for all experiments is included in the supplementary materials.

All experiments use 10 random seeds: {1825, 410, 4507, 4013, 3658, 5215, 6861, 6803, 7819, 8057}.

Compute resources. All experiments were conducted on a single Ubuntu 22.04.5 LTS machine running Linux kernel 5.15.0-161-generic. The machine has two Intel Xeon Gold 6248 CPUs at 2.50 GHz, providing 40 physical CPU cores and 80 hardware threads, 692 GiB of system memory, and four NVIDIA GeForce RTX 2080 Ti GPUs detected on the PCI bus. The NVIDIA driver and kernel module version was 555.42.02.

Existing assets and licenses. Our experiments use the Arcade Learning Environment (ALE) [6], OCAtari [10], CleanRL [14], Optuna [1], and the Deep W-Learning implementation associated with [23]. We use these assets only for research experiments and cite their original sources. The corresponding software licenses and terms of use are those distributed with the respective public repositories; we include the exact versions, repository URLs, and license names in the released supplementary code.

Table 5: Cat Feeder shared (training) environment parameters.

Parameter	Value
Grid size	30×30
Number of agents / targets	8
Target reward	50.0
Target expiry steps	200
Target expiry penalty	50.0
Max episode steps	2000
Moving targets	True
Direction change probability	0.1
Target move interval	5

Table 6: Cat Feeder multi-agent PPO hyperparameters (All-Pay and Winner-Pays).

Parameter	Value
Auction	
Bid upper bound	6
Bid penalty	0.1
Action window	5
Distance reward scale	0.6
Training	
Iterations	400
Parallel environments	4096
Steps per rollout	256
Minibatches	256
PPO epochs	4
Learning rate	2.5×10^{-4} (annealed)
Discount γ	0.99
GAE λ	0.95
Clip coefficient	0.05
Entropy coefficient	0.03
Value function coefficient	1.0
Max gradient norm	0.5
Network	
Actor hidden layers	$[128, 128, 128, 128]$
Critic hidden layers	$[256, 256, 256, 256]$
Target attention pooling	True
Target embedding dim	64
Target encoder hidden layers	$[64, 64]$

G.1 Cat Feeder Environment

G.2 Assault Environment

Table 7: Cat Feeder single-agent PPO baseline hyperparameters. Hyperparameters were tuned using Optuna with 40 trials. The nearest-target shaping variant is identical but uses a distance reward scale of 0.6 with shaping toward the nearest target.

Parameter	Value
Training	
Iterations	400
Parallel environments	4096
Steps per rollout	256
Minibatches	512
PPO epochs	8
Learning rate	1.74×10^{-4} (annealed)
Discount γ	0.963
GAE λ	0.970
Clip coefficient	0.327
Entropy coefficient	1.03×10^{-4}
Value function coefficient	1.076
Max gradient norm	0.840
Distance reward scale	0.0
Target attention pooling	False
Network (shared with multi-agent PPO)	
Actor hidden layers	$[128, 128, 128, 128]$
Critic hidden layers	$[256, 256, 256, 256]$

Table 8: Cat Feeder DWN baseline hyperparameters.

Parameter	Value
Training	
Total timesteps	5×10^8
Parallel environments	256
Discount γ	0.99
Replay buffer size	1×10^6
Batch size	256
Learning starts	1×10^5
Train frequency	256
W-network train delay	1×10^6
Target network update frequency	1000
Target network τ	1.0
Networks	
Q-network hidden layers	$[256, 256, 256, 256]$
W-network hidden layers	$[128, 128, 128]$
Q-network learning rate	1×10^{-4}
W-network learning rate	1×10^{-4}
Epsilon schedules	
Q ε start / min / decay	0.99 / 0.01 / 0.99
W ε start / min / decay	0.99 / 0.01 / 0.99

Table 9: Assault shared environment parameters.

Parameter	Value
Number of agents	3
Max enemies	3
Max episode steps	10000
Enemy destroy reward	10.0
Life loss penalty	10.0
Fire-while-hot penalty	8.0
Action window	15
Bid upper bound	2
Bid penalty	0.3

Table 10: Assault multi-agent PPO hyperparameters (All-Pay and Winner-Pays).

Parameter	Value
Training	
Iterations	150
Parallel environments	128
Steps per rollout	512
Minibatches	8
PPO epochs	8
Learning rate	1×10^{-4} (annealed)
Discount γ	0.99
GAE λ	0.95
Clip coefficient	0.05
Entropy coefficient	0.05
Value function coefficient	0.5
Max gradient norm	0.5
Target KL	0.02
Network	
Actor hidden layers	$[128, 128, 128, 128]$
Critic hidden layers	$[256, 256, 256, 256]$

Table 11: Assault single-agent PPO baseline hyperparameters. Entries list differences from the multi-agent PPO setting; all omitted parameters are shared. Target KL early stopping is disabled for this baseline.

Parameter	Value
Learning rate	2.5×10^{-4} (annealed)
Clip coefficient	0.1
Entropy coefficient	0.01
Target KL	—

Table 12: Assault DWN baseline hyperparameters.

Parameter	Value
Training	
Total timesteps	1×10^7
Parallel environments	8
Discount γ	0.99
Replay buffer size	5×10^5
Batch size	256
Learning starts	1×10^4
Train frequency	80
W-network train delay	1×10^5
Target network update frequency	1000
Target network τ	1.0
Networks	
Q-network hidden layers	$[256, 256, 256]$
W-network hidden layers	$[128, 128, 128]$
Q-network learning rate	1×10^{-4}
W-network learning rate	1×10^{-4}
Epsilon schedules	
Q ε start / min / decay	0.99 / 0.01 / 0.995
W ε start / min / decay	0.99 / 0.01 / 0.995

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 the paper's main contribution: an auction-based framework for adapting among local policies in multi-objective RL. The experimental claims are supported in Section 4, with scope and future directions discussed in the conclusion.

   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 discusses current scope limitations, including explicit objective decomposition, the scope of shared-policy transfer, limited experimental breadth, and the absence of safety guarantees in the heterogeneous feeding-and-charging experiment.

   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: The theoretical results in Section 2 state the assumptions for the bidding settings and include proofs in the main text.

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: Section 4 describes the environments, baselines, ablations, and evaluation protocol, while Appendix G provides seeds and hyperparameters needed to reproduce the main experiments.

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: The supplementary material includes the codebase and recorded rollouts, and Appendix G reports the seeds and experiment hyperparameters.

   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: Section 4 summarizes the experimental setup, environments, baselines, and evaluation metrics. Appendix G lists the environment parameters, PPO/DWN hyperparameters, random seeds, and tuning details.

   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: The learning curves in Figure 3 report the mean over 10 random seeds with shaded regions corresponding to one standard deviation.

   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: [No]

   Justification: The paper reports the hardware configuration, training lengths, rollout sizes, numbers of parallel environments, and other computational parameters in Appendix G. Wall-clock execution time and total compute budget were not recorded.

   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: The work uses simulated reinforcement-learning environments and does not involve human subjects, private data, or deployed systems. We are not aware of any aspect of the research that conflicts with the NeurIPS Code of Ethics.

   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: The paper is primarily methodological and experimental, and no deployment or human-facing application is studied.

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: The paper does not release high-risk generative models, scraped datasets, or other assets with an apparent high risk of misuse.

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: Appendix G credits the existing assets used in the experiments, cites their original sources, and states that the corresponding licenses, terms of use, versions, and repository URLs are included with the released supplementary code.

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: [Yes]

    Justification: The supplementary material includes the experimental codebase and recorded policy rollouts. The paper documents the environments, training settings, random seeds, and hyperparameters needed to interpret and use these 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: The paper does not involve crowdsourcing or research with human subjects.

    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: The paper does not involve crowdsourcing or research with human subjects, so IRB approval or equivalent review is not applicable.

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: The core method does not use LLMs as an important, original, or non-standard component.

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.