# MADPO: Multi-Agent Direct Preference Optimization for Cooperative LLM Agents

Zeyu Liang
Norheastern Uninersity
USA

Zihao Li
King's College London
UK

Xiao Liang
Ioawa State Universtiy
USA

Zhen Han
Amazon
UK

## Abstract

Coordinating multiple large language model (LLM) agents is commonly framed as multi-agent reinforcement learning, which assumes a verifiable team reward and requires costly online rollouts. These assumptions exclude many open-ended collaborative settings where humans can compare team outputs but cannot supply a reliable scalar reward. We propose Multi-Agent Direct Preference Optimization (MADPO), a direct preference-learning objective for cooperative LLM teams. MADPO sums per-agent policy log-ratios into a joint implicit reward and optimizes all agents with one Bradley–Terry sigmoid over the team margin, avoiding online rollouts, engineered rewards, and intermediate reward models. We show that this loss is exactly DPO applied to the factored joint policy, that its gradient gives every agent the same team-level weight, and that it reduces to independent per-agent DPO when preferences decouple across agents. Experiments on multi-dimensional evaluation and web shopping show that MADPO improves over independent DPO when agent decisions are sequentially coupled and remains comparable when agents act independently. A subgroup analysis further shows that joint optimization is most reliable when local preference signals agree with the overall team preference. Reward-model-based RL trained on the same preference data fails to match direct optimization at several times the cost.

**ACM Reference Format:**
Zeyu Liang, Zihao Li, Xiao Liang, and Zhen Han. 2026. MADPO: Multi-Agent Direct Preference Optimization for Cooperative LLM Agents. In . ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 Introduction

LLM-based multi-agent systems tackle complex tasks by assigning complementary roles to specialized agents, such as a search agent that gathers relevant information and a decision-making agent that acts on it. In such systems, team performance depends not only on the quality of each agent's output, but also on how well their outputs fit together. A query may be reasonable in isolation yet retrieve information that the downstream agent cannot use; conversely, a downstream decision may be locally sensible but fail because the necessary evidence was never retrieved. These failures highlight a central challenge in cooperative LLM systems: individually plausible behavior does not necessarily yield an effective joint solution.

Multi-agent reinforcement learning (MARL) provides a natural framework for addressing this challenge. By assigning a shared reward to the team, MARL methods can optimize agents toward a common objective rather than treating each agent independently [16, 34]. This approach is effective when the environment supplies a verifiable reward, such as unit-test success or a simulator score. However, many open-ended collaborative tasks do not admit a reliable scalar reward, even when humans can compare the quality of two candidate solutions. Moreover, MARL requires repeated online interaction with the environment, making training expensive for LLM agents.

A common alternative is to learn a reward model from preference data and then optimize the agents against the learned reward [21, 29]. Although this approach replaces manually designed rewards with human supervision, it retains a two-stage training pipeline: the reward model must first approximate the underlying preferences, after which the policies are optimized against this learned proxy. Errors in the reward model can therefore be amplified during policy optimization, particularly when the policies generate outputs outside the reward model's training distribution. In our experiments, reward models that accurately classify held-out preference pairs can still be exploited during reinforcement learning: their predicted rewards continue to increase while actual task performance remains nearly unchanged (Figure 2).

In many collaborative tasks, preferences over *joint* team outputs are available even when reliable scalar rewards or agent-level supervision are not. For example, a user may be able to compare two complete team solutions or prefer an expert demonstration over a sampled alternative without determining which individual agent caused the difference. This setting motivates the central question of this paper: *Can cooperative LLM agents be optimized directly from offline preferences over joint outputs, without learning an explicit reward model, interacting with the environment, or assigning preferences to individual agents?*

We introduce MADPO (Multi-Agent Direct Preference Optimization), an offline preference-optimization framework for cooperative LLM agents. MADPO applies Direct Preference Optimization (DPO) [23] to the factored joint policy of an agent team. For each preferred and dispreferred pair of joint outputs, it computes a policy log-ratio margin for every agent, aggregates these margins into


*Conference'17, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a single team-level margin, and optimizes all agents through one Bradley–Terry objective. MADPO therefore learns directly from team-level preferences, eliminating both the intermediate reward model and online policy rollouts.

Although the resulting objective is a simple extension of DPO, its optimization behavior differs fundamentally from applying DPO to each agent independently. Because the per-agent margins are combined before the Bradley–Terry sigmoid, the gradient received by every agent is scaled by the same team-level weight. Consequently, an agent's update depends on the preference margin of the entire team rather than only on its own local margin. This provides an offline preference-learning analogue of the shared team advantage used in cooperative policy-gradient methods. We additionally introduce an interpolation parameter $\alpha$ that connects pure joint optimization to independent per-agent DPO within a single objective.

The benefit of this shared signal depends on both the interaction structure and the compatibility of the agents' preference signals. When one agent's output changes the observations or decisions of another, independently optimizing the agents can reinforce locally reasonable but mutually incompatible behavior. In this coupled setting, the joint margin incorporates information from the complete team output that is absent from each local margin. In contrast, when the policies and preferences decompose across agents, we show that joint and independent DPO share the same population optimum. This equivalence additionally requires local preferences to be compatible with the team objective: when local and global preference labels conflict, a summed joint margin alone need not resolve the disagreement. These conditions establish the boundary studied in our experiments.

We make the following contributions:

- **Joint preference optimization for cooperative agents.** We formulate MADPO, which optimizes a factored team of LLM policies directly from preferences over joint outputs, without requiring an explicit reward model, engineered scalar rewards, or online environment interaction. The framework includes an $\alpha$-interpolated objective that recovers independent per-agent DPO as a special case.
- **Analysis of the joint learning signal.** Building on the standard DPO optimality result for the factored joint policy (Lemma 1), we show that the gradient of MADPO supplies all agents with a shared team-level weight (Eq. 6). We further establish the population-level relationship between direct preference optimization and reward-model-based reinforcement learning (Prop. 1) and characterize a decoupling condition under which joint training reduces to independent DPO (Prop. 2).
- **Evaluation across interaction and preference structures.** We evaluate MADPO on multi-dimensional preference judging and web shopping, covering both structurally independent and sequentially coupled agent systems. Under matched data and model capacity, MADPO consistently outperforms independent DPO in the coupled setting while preserving aggregate performance when agents act independently. A consistent–conflict analysis further isolates preference compatibility as a second factor governing the benefit of joint optimization.
- **Comparison with alternative coordination pipelines.** We compare MADPO with independent and joint MARL, reward-model-based reinforcement learning, parameter-sharing variants, and inference-time communication. The results show that direct joint optimization avoids the reward-model exploitation observed in the two-stage pipeline and achieves its coordination gains without additional online rollouts.

## 2 Related Work

*Prompt-level collaboration.* Multi-agent behavior can be elicited without training: sequential pipelines pass one agent's output into the next agent's prompt, and discussion methods iterate rounds of mutual critique [4, 9]. These methods can improve coordination, but they do so at inference time, adding rounds, tokens, and latency that grow with team size. We include sequential and discussion baselines to separate inference-time communication from training-time coordination, and show that MADPO internalizes coordination during training, avoiding the additional discussion rounds, tokens, and latency required by inference-time communication.

*MARL for LLM agents.* MAGRPO [16] trains collaborating LLM agents with group-relative policy optimization under a shared team reward, following the centralized-training paradigm of cooperative MARL [20, 34]. These methods define the relevant RL ceiling in our experiments when an oracle reward exists. Their limitation is that they require thousands of environment interactions and do not apply when no verifiable reward is available. MADPO transfers their central coordination mechanism—a shared team advantage—to the offline preference setting (Eq. 6). Related online recipes still require a computable reward at training time: decentralized actor-critic variants of MAGRPO [15], verifier-scored multi-turn co-training [22], sequential pioneer–observer coevolution [18], reward-curated SFT/DPO hybrids for multi-agent communication [1], and the general multi-agent reinforcement fine-tuning framework MARFT [14]; MADPO needs none.

*Direct preference optimization.* DPO [23] removes the reward model (RM) and RL loop from preference learning for a single policy; SimPO [19] and CPL [8] refine the objective, DMPO [27] extends it to multi-*turn* single-agent trajectories with a discounted step weighting, SDPO [11] instead selects key dialogue segments to optimize, and ETO [28] contrasts successful against failed exploration trajectories. These methods optimize one policy. MADPO extends direct preference optimization along an orthogonal axis: multiple policies whose updates are coupled through one team margin.

*Multi-agent preference optimization.* Mars-PO [17] and Flow-DPO [3] combine multiple agents with preference optimization, but pursue different goals: Mars-PO trains an ensemble of homogeneous solvers on one shared task, whereas Flow-DPO uses multi-component collaboration to produce reasoning data for a downstream model; neither coordinates heterogeneous roles toward one joint output. SysDPO [30] formulates a compound AI system as a directed acyclic graph and derives a system-level DPO objective by summing component-wise policy log-ratios within a single

Bradley–Terry loss. The pure joint objective in MADPO has the same mathematical form when specialized to a factored team of cooperative LLM agents. Our focus, however, is on the multi-agent learning properties of this objective. We analyze how the joint margin induces a shared team-level gradient signal, characterize when joint optimization reduces to independent per-agent DPO, and compare it empirically with independent preference optimization, MARL, and reward-model-based RL under matched data and training budgets. We also introduce an $\alpha$-interpolated objective that connects joint and independent optimization within a single framework. AMADPO [12] studies offline multi-agent preference-based RL on SMAC. Despite the name, AMADPO is two-stage: it first trains a multi-agent preference predictor—requiring per-agent local labels beyond global ones—and then optimizes the policy under importance weights derived from the predictor, reintroducing the model-error channel that direct methods avoid (the authors' own analysis documents a large gap between predicted and true rewards). MADPO instead applies a single joint-margin loss directly to preferences over joint outputs, without fitting a preference predictor. The global–local inconsistency studied by AMADPO concerns consistency between learned global and agent-level preference signals. Our UltraFeedback subgroup analysis examines a different source of disagreement: conflicts already present between observed dimension-level ratings and the overall preference label.

*Reward hacking and two-stage pipelines.* Optimizing against learned reward models is known to over-optimize proxy scores [6], and reward models that fit the preference data well can still generalize poorly off-distribution [24]; more broadly, whether two-stage RLHF or direct optimization performs better is governed by which model class is mis-specified [26]. MADPO adds an instance of this phenomenon: with the same preference pairs, direct optimization succeeds where the reward-model pipeline is hacked.

## 3 Problem Formulation

We model LLM collaboration as a fully cooperative decentralized partially observable Markov decision process (Dec-POMDP) [20], adopting the LLM Dec-POMDP formulation of Liu et al. [16]: a tuple $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}, \{\mathcal{A}_i\}, R, T, H\rangle$, where $\mathcal{I} = \{1, \ldots, N\}$ indexes pre-trained LLM agents; a state $s \in \mathcal{S}$ holds the task context $x$ and all text produced so far (our tasks involve no separate user state); a local observation $o_i \in \mathcal{O}_i$ is agent $i$'s role-specific prompt, a partial view of $s$; an action $y_i \in \mathcal{A}_i$ is a natural-language response; $T$ appends the joint action to the state (and, where applicable, executes it against external systems), embedding earlier outputs into subsequent prompts; and $H$ is a short episode horizon. All agents share one team reward $R$—which, departing from the reward-supervised setting of Liu et al. [16], is in our setting never observed.

Two interaction structures instantiate this model. In the *single-turn* case ($H$=1; e.g., multi-dimensional judging), all agents act simultaneously on observations derived from the initial state. In the *multi-turn* case ($H$=2), as in WebShop, an earlier agent changes the state before a later agent observes it; for example, the click agent sees the result page induced by the search agent's query. Agents are therefore coupled through the transition while execution remains decentralized, since each agent conditions only on its own observation. The joint policy factors as

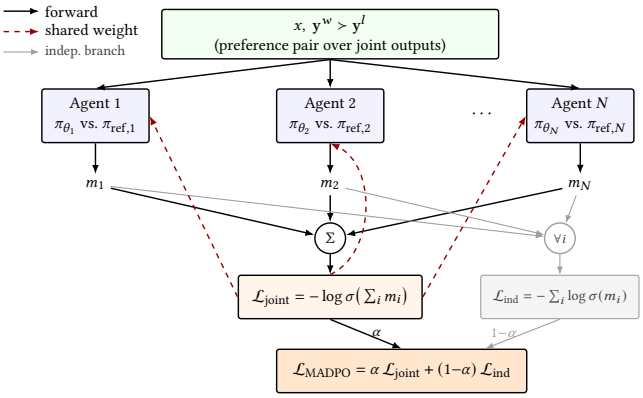


**Figure 1: The MADPO objective. Per-agent margins $m_i$ feed two branches: the *joint* branch (solid) sums them before one Bradley–Terry sigmoid, broadcasting the same team-level gradient weight to every agent, while the *independent* branch (gray) applies a per-agent sigmoid, weighting each agent by its own margin. The mixing weight $\alpha$ interpolates between them; $\alpha$=1 (pure joint) is our method and $\alpha$=0 recovers independent per-agent DPO.**

$$\pi_\theta(\mathbf{y} \mid x) \;=\; \prod_{i=1}^{N} \pi_{\theta_i}(y_i \mid o_i), \tag{1}$$

where $o_i$ may depend on teammates' earlier outputs in the multi-turn case (the product then denotes the corresponding chain-rule factorization).

Because $R$ is unobserved, we assume access only to preferences over joint outputs, $\mathcal{D} = \{(x, \mathbf{y}^w, \mathbf{y}^l)\}$ with $\mathbf{y}^w \succ \mathbf{y}^l$—the multi-agent analog of preference-based RL. The target team objective is KL-regularized return maximization,

$$\max_{\pi_\theta} \; \mathbb{E}_{\mathbf{y}\sim\pi_\theta}[r(x, \mathbf{y})] - \beta\, \mathbb{D}_{\mathrm{KL}}[\pi_\theta \,\|\, \pi_{\mathrm{ref}}], \tag{2}$$

with factored reference $\pi_{\mathrm{ref}}(\mathbf{y} \mid x) = \prod_i \pi_{\mathrm{ref},i}(y_i \mid o_i)$. For a single policy, DPO [23] shows the optimum of Eq. 2 satisfies $\pi^*(y \mid x) \propto \pi_{\mathrm{ref}}(y \mid x) \exp(r(x, y)/\beta)$, so the Bradley–Terry likelihood can be expressed purely in policy log-ratios. The question is what this substitution becomes for a factored team policy: given only preferences over *joint* outputs, what loss should update each agent?

## 4 MADPO

### 4.1 Joint-Margin Preference Optimization

For a factored team policy, the DPO log-ratio of the joint output decomposes as a sum of per-agent log-ratios. We therefore define the joint implicit reward as

$$r_\theta(x, \mathbf{y}) \;=\; \sum_{i=1}^{N} \beta \log \frac{\pi_{\theta_i}(y_i \mid o_i)}{\pi_{\mathrm{ref},i}(y_i \mid o_i)}, \tag{3}$$

and the per-agent margin $m_i = \beta \log \frac{\pi_{\theta_i}(y_i^w|o_i^w)}{\pi_{\mathrm{ref},i}(y_i^w|o_i^w)} - \beta \log \frac{\pi_{\theta_i}(y_i^l|o_i^l)}{\pi_{\mathrm{ref},i}(y_i^l|o_i^l)}$. The pure joint objective applies one Bradley–Terry sigmoid to the sum of these margins. To expose the boundary with independent

**Algorithm 1** MADPO

**Require:** preference data $\mathcal{D} = \{(x, \mathbf{y}^w, \mathbf{y}^l)\}$; base model $\pi_0$; mixing weight $\alpha$; temperature $\beta$; learning rate $\eta$; epochs $E$

1: **Initialize:** for each agent $i \in \{1, \dots, N\}$, policy $\pi_{\theta_i} \leftarrow \pi_0$ with trainable adapter $\theta_i$; reference $\pi_{\text{ref},i} \leftarrow \pi_0$ (frozen)
2: **for** epoch = 1 **to** $E$ **do**
3:  **for** each $(x, \mathbf{y}^w, \mathbf{y}^l) \in \mathcal{D}$ **do**
4:  build observations $o_i^w, o_i^l$ from $x$ (and, in multi-turn tasks, teammates' recorded outputs)
5:  $\ell_i^c \leftarrow \log \pi_{\text{ref},i}(y_i^w \mid o_i^w)$, $\ell_i^r \leftarrow \log \pi_{\text{ref},i}(y_i^l \mid o_i^l)$ ▷ all refs first, no grad
6:  **for** $i = 1$ **to** $N$ **do**
7:  $m_i \leftarrow \beta\big[\log \pi_{\theta_i}(y_i^w \mid o_i^w) - \ell_i^c\big] - \beta\big[\log \pi_{\theta_i}(y_i^l \mid o_i^l) - \ell_i^r\big]$
8:  **end for**
9:  $\mathcal{L} \leftarrow -\alpha \log \sigma\big(\sum_j m_j\big) - (1-\alpha) \sum_j \log \sigma(m_j)$
10:  $\theta_i \leftarrow \theta_i - \eta \nabla_{\theta_i} \mathcal{L} \quad \forall i$ ▷ one backward; shared weight $\sigma(-\sum_j m_j)$ reaches every agent
11:  **end for**
12: **end for**
13: **return** $\{\pi_{\theta_i}\}_{i=1}^N$

training, we optimize a weighted combination of a joint team-level objective and a component-wise auxiliary objective:

$$\mathcal{L}_{\text{joint}} = -\log \sigma(\textstyle\sum_i m_i), \quad \mathcal{L}_{\text{ind}} = -\sum_i \log \sigma(m_i), \tag{4}$$

$$\mathcal{L}_{\text{MADPO}} = \alpha\, \mathcal{L}_{\text{joint}} + (1-\alpha)\, \mathcal{L}_{\text{ind}}. \tag{5}$$

The joint term coordinates agents through the team margin, while the independent term prevents an individual agent from losing its learning signal when the team margin is already saturated.

Algorithm 1 gives the training procedure. Each offline pair requires $2N$ reference forwards and $N$ policy forwards, followed by one backward step that sends the shared sigmoid weight to every agent. There is no sampling from the current policy, no environment interaction, and no reward model. In multi-adapter implementations, reference log-probabilities must be computed with adapters disabled before any policy forward whose graph is retained for backpropagation (line 5; Appendix C). The only cross-agent quantity is the list of margins $m_i$, so agents can be implemented as separate models or as adapters on one shared base model.

## 4.2 Theoretical Properties

We build on one standard lemma and establish two properties specific to the multi-agent setting, with proofs given below.

Lemma 1 (MADPO optimality). *$\mathcal{L}_{\text{joint}}$ is the exact DPO objective of the joint policy $\pi_\theta$ in the factored class (Eq. 1), so when the team optimum is realizable its minimizer is the KL-regularized optimum of Eq. 2, $\pi_\theta^*(\mathbf{y} \mid x) \propto \pi_{\text{ref}}(\mathbf{y} \mid x) \exp\big(r(x, \mathbf{y})/\beta\big)$.*

*Proof.* For fixed $x$, the objective (2) over all distributions $p(\cdot \mid x)$ equals $-\beta\, \mathbb{D}_{\text{KL}}\big[p \,\|\, \frac{1}{Z(x)} \pi_{\text{ref}}\, e^{r/\beta}\big] + \beta \log Z(x)$ with $Z(x) = \sum_{\mathbf{y}} \pi_{\text{ref}}(\mathbf{y} \mid x)\, e^{r(x,\mathbf{y})/\beta}$, so it is maximized at $\pi_\theta^* \propto \pi_{\text{ref}}\, e^{r/\beta}$ (KL zero). Inverting gives $r(x, \mathbf{y}) = \beta \log(\pi_\theta^*/\pi_{\text{ref}}) + \beta \log Z(x)$; substituting into the Bradley–Terry likelihood $\sigma\big(r(x, \mathbf{y}^w) - r(x, \mathbf{y}^l)\big)$ cancels the $\beta \log Z(x)$ terms, and factoring $\pi_\theta, \pi_{\text{ref}}$ through (1) turns the log-ratio difference into $\sum_i m_i$, so the negative log-likelihood of the data is exactly $\mathcal{L}_{\text{joint}}$. Finally, if $r(x, \mathbf{y}) = \sum_i r_i(x, y_i)$ then $e^{r/\beta} = \prod_i e^{r_i/\beta}$, so $\pi_\theta^*$ factors into $\pi_i^* \propto \pi_{\text{ref},i}\, e^{r_i/\beta}$ and lies in the factored class, where it is attained. ∎

This is the standard single-policy DPO/RLHF equivalence [7, 23], stated here for the factored joint policy and used as the basis for the results that follow. The DPO derivation uses only the closed form of the KL-regularized optimum and partition-function cancellation under the Bradley–Terry substitution; neither step depends on the policy's internal structure, so the single-policy argument transfers verbatim, with the joint log-ratio decomposing into $\sum_i m_i$. SysDPO [30] derives the same loss and an exact preference-oracle alignment under a *uniform* reference and a full-support assumption; our statement instead retains an arbitrary frozen reference $\pi_{\text{ref}}$, the KL-regularized form standard in DPO practice. Realizability requires the team reward to be representable in the factored class, e.g., additive across agents.

*Shared credit assignment.* Differentiating the joint loss gives, for each agent, a one-line identity

$$\nabla_{\theta_i} \mathcal{L}_{\text{joint}} = -\,\sigma(-\textstyle\sum_j m_j)\; \nabla_{\theta_i} m_i, \tag{6}$$

so every agent receives the *same* scalar weight $\sigma(-\sum_j m_j)$, set by the team margin—the offline analog of a shared-advantage policy gradient—whereas $\mathcal{L}_{\text{ind}}$ weights agent $i$ by its own $\sigma(-m_i)$. Elementary as it is, this identity is the single mechanism that separates joint from independent training and the quantity our $\alpha$-mix interpolates (Eq. 5). It places MADPO in the lineage of counterfactual and difference-reward credit assignment in cooperative MARL [5, 31], and alongside concurrent credit assignment for multi-agent LLM collaboration [13]—though ours falls directly out of differentiating the DPO objective, with no credit-assignment module to fit. The identity follows in one line: with $u = \sum_j m_j$ and $\frac{d}{du}[-\log \sigma(u)] = -(1 - \sigma(u)) = -\sigma(-u)$, and since each $m_j$ depends on $\theta_j$ alone ($\nabla_{\theta_i} u = \nabla_{\theta_i} m_i$), the chain rule gives (6); applying the same computation term-wise to $\mathcal{L}_{\text{ind}} = \sum_j -\log \sigma(m_j)$ yields $-\sigma(-m_i)\nabla_{\theta_i} m_i$—the same direction, weighted by the agent's own margin instead of the team's.

Both gradients move agent $i$ along $\nabla_{\theta_i} m_i$, toward higher relative likelihood of the chosen output; they differ only in the scalar multiplier. Under the joint loss, an agent whose own margin is already large keeps receiving gradient while the *team* margin is unsaturated, so it cannot opt out of team failures. Empirically this appears as stability: independent training collapses in several seeded runs, joint training does not (Sec. 6). Conversely, one agent alone can saturate the team margin, letting the shared weight vanish for all agents; this lazy-agent risk motivates the $\alpha$-mix in Eq. 5.

Proposition 1 (RM pipeline shares MADPO's optimum). *Assume the conditions of Lemma 1: preferences follow a Bradley–Terry model with reward additive across agents, $r(x, \mathbf{y}) = \sum_i r_i(x, y_i)$, so the optimum is realizable in the factored class (Eq. 1). Let $\hat{r}$ be the population minimizer of the Bradley–Terry reward-modeling loss on joint outputs, and let $\hat{\pi}$ maximize the KL-regularized objective (Eq. 2) with $\hat{r}$ in place of $r$. Then $\hat{\pi}$ coincides with the population minimizer of $\mathcal{L}_{\text{joint}}$. Thus, in the population and under these assumptions, the pipeline has no better optimum to reach; any finite-sample gap is driven by reward-model estimation error and by over-optimization*

*against the fixed proxy [6], both of which direct optimization structurally avoids (cf. Fig. 2).*

*Proof.* The Bradley–Terry reward-modeling loss depends on $\hat{r}$ only through pairwise differences $\hat{r}(x, \mathbf{y}^w) - \hat{r}(x, \mathbf{y}^l)$, which are invariant to $\hat{r} \mapsto \hat{r} + c(x)$; and since $-\log\sigma$ is strictly convex, the loss is minimized iff those differences match $r$'s on all pairs, so the minimizer set is exactly $\{\, r + c(x) \,\}$ [7]. For any such $\hat{r} = r + c(x)$, the KL-regularized optimum of Lemma 1 is $\hat{\pi} \propto \pi_{\text{ref}}\, e^{(r+c(x))/\beta} = e^{c(x)/\beta}\pi_{\text{ref}}\, e^{r/\beta}$, and the constant $e^{c(x)/\beta}$ cancels on normalization, so $\hat{\pi} = \pi^*_\theta$. By Lemma 1 $\pi^*_\theta$ is also the minimizer of $\mathcal{L}_{\text{joint}}$, which under additivity lies in the factored class where both routes attain it. ∎

The argument rests on two observations. Preference data determine the reward only up to a prompt-dependent offset, and the KL-regularized optimum is invariant to precisely that offset, which cancels in the normalizing constant. The RM stage therefore passes no information to the RL stage beyond what the preference data already determine, and both routes recover the same optimal policy. Because this reasoning concerns optimal policies rather than a particular parameterization, Lemma 1 transfers it unchanged from the single-policy setting [7, 23] to the factored team policy. The realizability assumptions are not merely technical: when the reward class does not contain the true reward, or the policy class does not contain the optimal policy, the equivalence breaks down and either route can strictly outperform the other, depending on which class is mis-specified [26]; the proposition therefore states its conditions explicitly. One genuinely multi-agent distinction remains: after fitting $\hat{r}$, the pipeline must still decide how to distribute the team reward across agents during RL—our joint and independent $\tilde{R}$ baselines instantiate the two natural choices—whereas MADPO embeds this decision in the loss itself (Eq. 6).

Proposition 2 (Decoupled tasks). *Suppose each agent's preference data are generated independently according to a Bradley–Terry model with private reward $r_i(x, y_i)$, and the joint preference pairs are component-wise consistent: for every agent $i$, $y_i^w \succ_i y_i^l$. Suppose further that the team reward is additive, $r(x, \mathbf{y}) = \sum_i r_i(x, y_i)$. Then the population minimizers of $\mathcal{L}_{\text{joint}}$ and $\mathcal{L}_{\text{ind}}$ coincide at the per-agent optima $\pi_i^* \propto \pi_{\text{ref},i} \exp(r_i/\beta)$.*

*Proof.* $\mathcal{L}_{\text{ind}}$ is a sum of $N$ single-policy DPO losses on disjoint parameter sets, each minimized at $\pi_i^* \propto \pi_{\text{ref},i} e^{r_i/\beta}$ by single-policy DPO consistency, so the sum is minimized at $\{\pi_i^*\}$. Applying Lemma 1 with the additive reward, the team optimum factors into the same $\pi_i^*$, so both losses are minimized by $\prod_i \pi_i^*$. ∎

This proposition gives a data-level boundary for MADPO. When agent-level preferences are generated independently, are component-wise consistent, and correspond to an additive team reward, the joint margin does not change the population optimum relative to independent DPO. Finite-sample optimization may still produce numerical differences, but the joint objective has no additional coordination problem to solve.

The component-wise consistency condition is essential. Reward additivity alone does not guarantee that every component of the preferred joint output is locally preferred: a team-level comparison may favor one joint output even when some of its components are locally worse. When local and global preferences conflict, the joint and independent objectives need not share the same optimum, and a summed margin provides no explicit mechanism for deciding which local signal should dominate.

| | UltraFeedback | WebShop |
|---|---|---|
| Agents | 4 (judges) | 2 (search + click) |
| Interaction | independent | sequentially coupled |
| Local/global labels | mixed | aligned by construction |
| Base model | Qwen3-4B | Qwen3-4B |
| Pairs (train) | 1067 | 972 |
| Chosen source | UF ratings | human demos |
| Test set | 187 pairs | 100/300 goals |
| Metric | accuracy | reward, success |
| Oracle $R^*$ | ***none*** | env. reward |

**Table 1: The two evaluation domains. "Interaction" describes whether one agent's output affects another agent's observation. "Local/global labels" describes whether agent-level preferences agree with the team-level target. "Oracle $R^*$" denotes the reward available to RL baselines; UltraFeedback has no directly available scalar oracle reward.**

Our experiments expose both boundaries. WebShop introduces sequential interaction dependence while keeping the demonstrated trajectory component-wise dominant over the sampled alternative. UltraFeedback removes interaction dependence between judges while providing both component-wise consistent pairs and pairs in which dimension-level ratings disagree with the overall preference. The subgroup analysis in Sec. 6 therefore complements the population result by revealing how preference compatibility shapes joint optimization on held-out data.

## 4.3 Implementation

In our implementation, all agents share one frozen base model and differ only in LoRA adapters [10]. The reference policy is the same base model with adapters disabled, so no second model copy is required. Two details are important in this multi-adapter setting. First, reference log-probabilities must be computed *before* any policy forward pass whose computation graph is kept for the backward step. Second, per-agent immediate backward passes, using a first-order exact surrogate for the joint coefficient, bound peak memory by a single agent's activation graph. Appendix C details both issues, including a silent zero-gradient failure mode in a popular adapter library.

# 5 Experimental Setup

We evaluate MADPO on two cooperative LLM-agent domains: preference judging and web shopping. Together they span single- and multi-turn interaction, structurally independent and sequentially coupled agent roles, aligned and conflicting local preference signals, and settings with and without an oracle reward. Dataset construction and additional results are in the supplement.

*Domains.* Table 1 summarizes the domains.

- *UltraFeedback* [2]: dimension-specialist judges independently score two candidate responses. Each judge is trained on pairs

oriented according to its own dimension-level rating. The team verdict aggregates the per-agent implicit rewards with a logistic head, and accuracy is measured against the overall human preference. The judges do not observe or condition on one another, so the interaction structure is independent. However, the dimension-level ratings do not always agree with the overall label. We therefore divide the 187 test pairs into 143 *consistent* examples, for which every dimension with a strict rating difference prefers the overall winner, and 44 *conflict* examples, for which at least one dimension prefers the overall loser.

- *WebShop* [33]: a search agent writes the query and a click agent emits a full purchase plan over the titled result page; chosen behavior comes from human demonstrations, rejected from base-model samples; the metric is the official purchase reward. The two agents are tightly *coupled*: the click decision depends on what the search returns.

*Baselines.* We compare against three baseline families.

- *Inference-time*: zero-shot single pass, naive concatenation, sequential pipeline, and one-round discussion [16].
- *Preference-trained*: single-model DPO (one shared policy, independent sigmoids); independent DPO ($\alpha$=0); MADPO ($\alpha$=1). No public implementation of AMADPO is available, precluding a direct empirical comparison. We exclude SysDPO because it targets hierarchical DAG pipelines, unlike our cooperative role-based setting. Moreover, with observed intermediate outputs, SysDPO-Direct is equivalent to MADPO with $\alpha$=1.
- *RL-trained*: GRPO variants [25] with the oracle reward $R^*$ where it exists (independent learners; MAGRPO-style joint with shared group-relative advantage [16]), and the two-stage pipeline $\tilde{R}$ that fits a Bradley–Terry reward model on the *same* preference pairs and runs GRPO against it.

All trained methods share the base model (Qwen3-4B; 32), *per-adapter* LoRA rank ($r$=32), and data; joint and independent variants have identical total trainable parameters, while the single-model variant deliberately shares one adapter across roles (Sec. 6). The DPO temperature is set per domain ($\beta$=0.15 on UltraFeedback, $\beta$=0.5 on WebShop, whose near-optimal demonstrations require a stronger reference anchor) and applied uniformly to *all* DPO variants within a domain. All trained runs use three seeds on a single NVIDIA A40 GPU; training costs are reported separately.

## 6 Results

Tables 2 and 3 report the main results. Two distinct factors govern the value of joint optimization. Interaction dependence determines whether the agents can benefit from a shared team-level training signal, while local–global preference consistency determines whether their individual margins point toward a compatible team objective. The following subsections analyze these two boundaries and compare direct preference optimization with reward-model-based RL, parameter sharing, and inference-time coordination.

| | 100 goals | | 300 goals | | |
|---|---|---|---|---|---|
| Method | Reward | Succ. | Reward | Succ. | Cost |
| *Inference-time (no training)* | | | | | |
| Zero-shot | 55.5 | 4.0 | 52.8 | 2.7 | N/A |
| Discussion † | 43.8 | 5.0 | 42.7 | 2.7 | N/A |
| *RL-trained (oracle or fitted reward)* | | | | | |
| MAGRPO $R^*$ | **75.0±1.3** | **43.0±1.7** | **73.9±0.2** | **42.2±1.2** | 2.7h |
| GRPO $R^*$ | 63.5±7.5 | 19.0±22.5 | 61.7±9.1 | 16.8±22.7 | 2.8h |
| MAGRPO $\tilde{R}$ | 54.7±0.3 | 6.3±2.5 | 51.7±0.5 | 4.2±1.8 | 2.8h‡ |
| GRPO $\tilde{R}$ | 56.8±1.7 | 10.1±3.2 | 52.6±0.8 | 6.8±2.2 | 2.8h‡ |
| *Preference-trained* | | | | | |
| Single-model DPO | 54.2±5.6 | 23.7±7.1 | 53.0±5.1 | 19.4±6.9 | ∼0.5h |
| MADPO ($\alpha$=0) | 55.0±2.9 | 21.7±5.5 | 52.5±3.4 | 18.0±4.6 | ∼0.5h |
| MADPO ($\alpha$=1) | **58.6±1.2** | **24.0±5.2** | **56.3±2.5** | **19.7±5.8** | ∼0.5h |

**Table 2: WebShop test results. *Trained* rows report mean±std over three seeds; bold marks the best value within each block. Only MAGRPO $R^*$ uses the oracle reward. Cost is wall-clock training time on one NVIDIA A40; The BT RM cannot be evaluated on reward/success directly because it only scores trajectories and takes no actions itself. †Discussion adds a query-revision turn. ‡Plus a ∼15-minute reward-model fit.**

| Method | Acc. | Consist. | Confl. | Cost |
|---|---|---|---|---|
| *Inference-time (no training)* | | | | |
| Zero-shot (single judge) | 70.1 | 75.9 | 51.1 | N/A |
| Naive concat | 86.9 | 94.4 | 62.5 | N/A |
| Sequential | 82.9 | 90.9 | 56.8 | N/A |
| Discussion | 84.0 | 91.6 | 59.1 | N/A |
| *RL-trained (two-stage $\tilde{R}$ pipeline)* | | | | |
| BT RM (single head) | 82.4±1.6 | 91.6±2.0 | 52.3±4.9 | ∼1h |
| BT RM (multi head) | 80.7±0.0 | 90.9±0.6 | 47.7±1.9 | ∼1h |
| MAGRPO $\tilde{R}$ | 58.1±3.8 | 61.2±3.3 | 48.0±5.2 | 5.6h |
| GRPO $\tilde{R}$ | 60.6±5.4 | 63.0±5.1 | 52.8±5.2 | 5.6h |
| *Preference-trained* | | | | |
| Single-model DPO | 85.0±1.4 | 91.1±1.5 | 65.2±2.6 | ∼1.5h |
| MADPO ($\alpha$=0) | 84.9±0.3 | 90.7±0.4 | 65.9±0.0 | 2.5h |
| **MADPO** ($\alpha$=1) | **85.4±1.1** | **92.3±0.7** | 62.9±2.6 | 2.5h |

**Table 3: UltraFeedback test results. *Consist./Confl.* split the 187 test pairs by component-wise consistency: on 143 pairs every dimension with a strict gold-rating difference prefers the overall winner (*consistent*); on the remaining 44 at least one dimension prefers the overall loser (*conflict*). *Trained* rows report mean±std over three seeds. Single-model DPO trains one judge on the overall preference with no dimension decomposition.**

### 6.1 Joint vs. Independent Training

On WebShop, where the click agent acts on the page produced by the search agent's query, joint preference optimization consistently

improves over independent per-agent DPO. On the 300-goal test set, increasing $\alpha$ from 0 to 1 raises the average reward from 52.5 to 56.3 and success from 18.0% to 19.7%. The reward improvement appears in every seed and is significant under a paired bootstrap over the 3×300 evaluated goals ($p<0.001$). Together with the consistent reward improvement, this result supports the mechanism in Eq. 6: each agent continues to receive a learning signal based on the margin of the complete search–click trajectory rather than only its own local margin.

The oracle-reward RL baselines exhibit a stronger version of the same pattern. MAGRPO with a shared team advantage reaches 42.2±1.2% success on 300 goals, whereas independent GRPO reaches 16.8±22.7%. The large variance of independent GRPO indicates that its performance is driven by one successful seed, while the joint learner converges consistently. Although oracle RL is not directly comparable to offline preference optimization, this result provides complementary evidence that a shared team-level signal is particularly important in the coupled WebShop pipeline.

On UltraFeedback, the four dimension-specific judges do not condition on one another. At the aggregate level, joint and independent training are nearly indistinguishable: $\alpha$=1 obtains 85.4±1.1% accuracy, compared with 84.9±0.3% for $\alpha$=0. This result indicates that the joint margin provides no clear overall advantage when the agents' decisions are structurally independent. A finer sweep in Appendix B similarly shows no reliable benefit from intermediate values of $\alpha$.

The aggregate comparison, however, hides opposite trends across the preference-consistency subsets in Table 3. On the 143 consistent pairs, joint MADPO reaches 92.3±0.7%, compared with 90.7±0.4% for independent DPO. On the 44 conflict pairs, the ordering reverses: joint MADPO obtains 62.9±2.6%, while independent DPO reaches 65.9±0.0%. This reversal identifies preference compatibility as a substantive boundary: the shared margin is most effective when local signals support a coherent team-level preference.

The conflict subset is difficult for all evaluated methods. For example, naive concatenation falls from 94.4% on consistent pairs to 62.5% on conflict pairs, while single-judge DPO falls from 91.1±1.5% to 65.2±2.6%. These results indicate that local–global disagreement is not a failure mode unique to MADPO. However, the shared joint margin does not resolve it: summing the agents' margins provides no mechanism for determining which local preference should dominate when the signals point in different directions.

Taken together, WebShop and UltraFeedback establish a two-axis account of joint preference optimization. Interaction dependence creates the need for a shared team-level training signal, while preference compatibility determines whether the agents' local margins reinforce a coherent team objective. MADPO delivers consistent gains in the coupled setting, preserves aggregate performance when interaction is independent, and exposes local–global conflict as a distinct challenge rather than conflating it with coordination.

## 6.2 Direct Optimization vs. the Reward-Model Pipeline

Direct preference optimization substantially outperforms the two-stage alternative of fitting a reward model and then optimizing agents against its predictions. On WebShop, joint MADPO reaches 19.7±5.8% success on 300 goals, while MAGRPO trained against the learned reward $\tilde{R}$ reaches only 4.2±1.8%, close to the 2.7% zero-shot baseline. This failure does not arise from poor in-distribution preference classification: the WebShop reward model achieves perfect held-out pairwise accuracy before it is used for policy optimization.

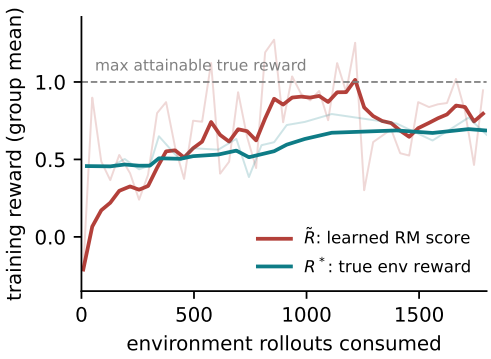


**Figure 2: Reward hacking on WebShop. RM-family against the learned reward model $\tilde{R}$ drives the RM score past the true-reward ceiling while success stays at base level; oracle-reward training remains bounded and reaches 42.2±1.2% success.**

Figure 2 explains the discrepancy. During GRPO training, the predicted reward continues to increase and eventually exceeds the maximum attainable true environment reward, while actual task performance remains nearly flat. The reward model therefore ranks held-out preference pairs correctly but fails on the policy-generated trajectories encountered during optimization, matching the reward-model generalization gap described by Razin et al. [24].

The same pattern is even clearer on UltraFeedback. The fitted reward models obtain approximately 81–82% held-out accuracy, but policies optimized against them fall to 58.1±3.8% and 60.6±5.4%. In contrast, directly preference-trained methods remain near 85%. These results do not imply that every reward-model pipeline must fail; rather, they show that accurate pairwise validation performance is insufficient to guarantee a reliable optimization target. MADPO avoids this additional failure mode by optimizing the observed preferences directly without materializing an explicit reward function.

## 6.3 $\alpha$ Sweep

Table 2 and 3 report the two endpoints of the mixing weight, $\alpha$=1 (pure joint) and $\alpha$=0 (independent). Table 4 gives a finer sweep on both domains, with three seeds per setting, and the two domains behave exactly as the coupling analysis predicts. On WebShop (coupled), pure joint optimization ($\alpha$=1) achieves the highest reward and success, while every interpolated objective underperforms the joint endpoint; in particular, $\alpha$=0.25 is substantially less stable, with a reward standard deviation of 13.9. On UltraFeedback, where the judges are interaction-independent, aggregate performance is insensitive to the mixing weight: all five settings lie within 84.3–85.4% accuracy, with differences comparable to cross-seed variation. This result is consistent with Proposition 2: under factorwise preference supervision, joint and independent objectives share the same

| | $\alpha$=0 | 0.25 | 0.5 | 0.75 | $\alpha$=1 |
|---|---|---|---|---|---|
| WebShop Reward | 52.5±3.4 | 32.9±13.9 | 48.7±3.7 | 48.5±2.4 | **56.3±2.5** |
| WebShop Success | 18.0±4.6 | 11.1±5.4 | 15.1±3.7 | 13.2±2.6 | **19.7±5.8** |
| UltraFeedback Acc. | 84.9±0.3 | 84.3±0.3 | 84.5±0.4 | 85.2±0.3 | **85.4±1.1** |

**Table 4: Sweep of the mixing weight $\alpha$.**

population optimum, leaving little aggregate benefit for interpolated objectives. Together, the two sweeps show that the shared joint margin is not one endpoint of a sensitive interpolation: where interaction coupling matters, it is the strongest and most reliable objective; where coupling is absent, aggregate performance is largely insensitive to $\alpha$. Accordingly, we treat the independent term as an ablation rather than a regularizer that requires tuning.

## 6.4 Parameter-Sharing Baseline

The single-model DPO baseline tests whether the WebShop gain instead comes from sharing parameters across roles. It obtains 53.0±5.1 reward and 19.4±6.9% success on 300 goals, compared with 56.3±2.5 and 19.7±5.8% for joint MADPO. Parameter sharing therefore does not explain the improvement in reward produced by the joint objective. The main difference is whether the search and click margins are combined before applying the Bradley–Terry loss.

## 6.5 Training Cost

MADPO removes both environment interaction and reward-model training. On UltraFeedback, single-judge DPO requires approximately 1.5 hours and multi-agent MADPO 2.5 hours. The two-stage RL pipeline, however, requires approximately one hour to fit the reward model followed by 5.6 hours of policy optimization. It is therefore more than twice as expensive as MADPO while producing substantially lower accuracy.

On WebShop, oracle-reward MAGRPO remains the strongest method, reaching 42.2±1.2% success. That comparison represents a performance ceiling rather than the supervision regime targeted by this work, because it assumes access to the true environment reward throughout training. When only offline team preferences are available, MADPO provides a simpler alternative: it requires no online rollouts, engineered reward, or separately trained reward model.

## 6.6 Inference-Time Coordination

The inference-time baselines distinguish the value of combining multiple roles from the value of adding further communication rounds. On UltraFeedback, aggregating several dimension-specific judgments is highly effective: naive concatenation reaches 86.9% accuracy, compared with 70.1% for a single zero-shot judge. More structured interaction does not improve further, however; sequential evaluation and discussion obtain 82.9% and 84.0%, respectively. Because the judges operate on separate dimensions, additional communication contributes little beyond aggregating their outputs.

On WebShop, the environment already provides the necessary communication channel: the click agent observes the result page induced by the search agent's query. Adding a query-revision round does not improve success and reduces reward from 52.8 to 42.7 on the 300-goal evaluation. The extra round can therefore perturb an initially useful query without supplying new task information.

These findings clarify the role of MADPO. It does not eliminate the sequential interaction required by environments such as WebShop; instead, it learns coordination through the joint training objective without adding extra discussion or revision rounds at inference time.

# 7 Conclusion

We introduced MADPO, a direct preference-optimization framework that trains cooperative LLM agents from offline preferences over joint outputs. By applying DPO to the factored joint policy, MADPO converts per-agent policy log-ratios into a single team margin and couples all agent updates through a shared team-level weight, without requiring an engineered reward, an intermediate reward model, or online environment interaction. Our analysis establishes the connection between joint preference optimization and KL-regularized team objectives, characterizes its shared-gradient mechanism, and identifies the conditions under which joint and independent DPO recover the same population optimum.

The experiments validate this account across complementary coordination structures. On sequentially coupled WebShop, joint MADPO improves over independent DPO in every seed, while the UltraFeedback analysis shows that it preserves strong aggregate performance when judges act independently and reaches 92.3% accuracy when local preferences align with the overall team target. The consistent–conflict decomposition further separates preference incompatibility from interaction-based coordination, providing a sharper account of when joint optimization is effective. Across both domains, direct preference optimization decisively outperforms reward-model-based RL trained on the same preference data, while avoiding the proxy exploitation and additional training cost of the two-stage pipeline. Together, these results establish joint direct preference optimization as a principled and practical foundation for training cooperative LLM systems when team-level preferences are available but reliable rewards are not.

## References


[1] Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2025. Optima: Optimizing Effectiveness and Efficiency for LLM-Based Multi-Agent System. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025 (Findings of ACL, Vol. ACL 2025)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 11534–11557. doi:10.18653/V1/2025.FINDINGS-ACL.601

[2] Ganqu Cui, Lifan Yuan, Ning Ding, et al. 2023. UltraFeedback: Boosting Language Models with High-Quality Feedback. *arXiv preprint arXiv:2310.01377* (2023).

[3] Yihe Deng and Paul Mineiro. 2024. Flow-DPO: Improving LLM Mathematical Reasoning through Online Multi-Agent Learning. *CoRR* abs/2410.22304 (2024). arXiv:2410.22304 doi:10.48550/ARXIV.2410.22304

[4] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving Factuality and Reasoning in Language Models through Multiagent Debate. In *International Conference on Machine Learning*.

[5] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 2974–2982. doi:10.1609/AAAI.V32I1.11794

[6] Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling Laws for Reward Model Overoptimization. In *International Conference on Machine Learning*.

[7] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A General Theoretical Paradigm to Understand Learning from Human Preferences. In *Artificial Intelligence and Statistics*, Vol. 238. 4447–4455. https://mlanthology.org/aistats/2024/gheshlaghiazar2024aistats-general/

[8] Joey Hejna, Rafael Rafailov, Harshit Sikchi, et al. 2023. Contrastive Preference Learning: Learning from Human Feedback without RL. *arXiv preprint arXiv:2310.02054* (2023).

[9] Sirui Hong, Mingchen Zhuge, Jonathan Chen, et al. 2024. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In *International Conference on Learning Representations*.

[10] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[11] Aobo Kong, Wentao Ma, Shiwan Zhao, Yongbin Li, Yuchuan Wu, Ke Wang, Xiaoqian Liu, Qicheng Li, Yong Qin, and Fei Huang. 2025. SDPO: Segment-Level Direct Preference Optimization for Social Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 12409–12423. doi:10.18653/V1/2025.ACL-LONG.607

[12] Qian Kou, Mingyang Li, Zeyang Liu, Long Qian, Zhuoran Chen, Lipeng Wan, Xingyu Chen, and Xuguang Lan. 2025. Offline Multi-Agent Preference-Based Reinforcement Learning with Agent-aware Direct Preference Optimization. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1181–1190.

[13] Zhongyi Li, Wan Tian, Jinju Chen, Huiming Zhang, Yang Liu, Yikun Ban, and Fuzhen Zhuang. 2026. Counterfactual Credit Policy Optimization for Multi-Agent Collaboration. *CoRR* abs/2603.21563 (2026). arXiv:2603.21563 doi:10.48550/ARXIV.2603.21563

[14] Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. 2025. MARFT: Multi-Agent Reinforcement Fine-Tuning. *CoRR* abs/2504.16129 (2025). arXiv:2504.16129 doi:10.48550/ARXIV.2504.16129

[15] Shuo Liu, Tianle Chen, Ryan Amiri, and Christopher Amato. 2026. Learning Decentralized LLM Collaboration with Multi-Agent Actor Critic. In *Forty-third International Conference on Machine Learning*. https://openreview.net/forum?id=k9OuSat5IR

[16] Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. 2026. LLM Collaboration with Multi-Agent Reinforcement Learning. In *Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, Sven Koenig, Chad Jenkins, and Matthew E. Taylor (Eds.). AAAI Press, 32150–32158. doi:10.1609/AAAI.V40I38.40487

[17] Xiaoxuan Lou, Chaojie Wang, and Bo An. 2024. Mars-PO: Multi-Agent Reasoning System Preference Optimization. *CoRR* abs/2411.19039 (2024). arXiv:2411.19039 doi:10.48550/ARXIV.2411.19039

[18] Hao Ma, Tianyi Hu, Zhiqiang Pu, Boyin Liu, Xiaolin Ai, Yanyan Liang, and Min Chen. 2024. Coevolving with the Other You: Fine-Tuning LLM with Sequential Cooperative Multi-Agent Reinforcement Learning. *CoRR* abs/2410.06101 (2024). arXiv:2410.06101 doi:10.48550/ARXIV.2410.06101

[19] Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple Preference Optimization with a Reference-Free Reward. In *Advances in Neural Information Processing Systems*.

[20] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer.

[21] Long Ouyang et al. 2022. Training Language Models to Follow Instructions with Human Feedback. In *Advances in Neural Information Processing Systems*.

[22] Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. 2025. MAPoRL: Multi-Agent Post-Co-Training for Collaborative Large Language Models with Reinforcement Learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Vienna, Austria, 30215–30248. doi:10.18653/v1/2025.acl-long.1459

[23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in Neural Information Processing Systems*.

[24] Noam Razin, Yong Lin, Jiarui Yao, and Sanjeev Arora. 2026. Why is Your Language Model a Poor Implicit Reward Model?. In *International Conference on Learning Representations (ICLR)*.

[25] Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024).

[26] Ruizhe Shi, Minhak Song, Runlong Zhou, Zihan Zhang, Maryam Fazel, and Simon S. Du. 2026. Understanding the Performance Gap in Preference Learning: A Dichotomy of RLHF and DPO. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*.

[27] Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. 2024. Direct Multi-Turn Preference Optimization for Language Agents. In *Proceedings of EMNLP*. 2312–2324.

[28] Yifan Song, Da Yin, Xiang Yue, et al. 2024. Trial and Error: Exploration-Based Trajectory Optimization for LLM Agents. In *Proceedings of ACL*.

[29] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2020/hash/1f89885d556929e98d3ef9b86448f951-Abstract.html

[30] Xiangwen Wang, Yibo Jacky Zhang, Zhoujie Ding, Katherine Tsai, Haolun Wu, and Sanmi Koyejo. 2025. Aligning Compound AI Systems via System-level DPO. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=6IuURCuooO

[31] David H. Wolpert and Kagan Tumer. 2001. Optimal Payoff Functions for Members of Collectives. *Adv. Complex Syst.* 4, 2-3 (2001), 265–280. doi:10.1142/S0219525901000188

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

[33] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*.

[34] Chao Yu, Akash Velu, Eugene Vinitsky, et al. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*.

# A Prompts and Hyperparameters

## A.1 Training Configuration

All trained methods use the same *per-adapter* LoRA configuration and optimizer, so that table differences reflect the training objective rather than per-agent capacity. Each adapter uses rank $r$=32, $\alpha$=64, dropout 0.05, applied to the {q,k,v,o} projection matrices; all other weights are frozen. The number of adapters equals the number of agents (two on WebShop, four on UltraFeedback), so joint and independent variants have identical total trainable parameters, while the single-model variant shares one adapter across roles—a deliberate parameter-sharing ablation, not a capacity handicap. We optimize with AdamW and a cosine schedule with 10% warmup, gradient clipping at 1.0. Preference (DPO/MADPO) runs use $E$=2 epochs; online-RL runs use $E$=1 epoch over the training goals. The DPO temperature is set per domain and applied uniformly to every DPO variant within that domain: $\beta$=0.15 on UltraFeedback, $\beta$=0.5 on WebShop (the near-optimal WebShop demonstrations make the rejected samples close to the chosen ones, so a stronger reference anchor is needed to avoid degenerate solutions). The main comparisons use the two endpoints of the MADPO mixing weight: $\alpha$=1 for the pure joint margin and $\alpha$=0 for independent per-agent margins. Appendix C additionally evaluates $\alpha \in \{0.25, 0.5, 0.75\}$ to test whether interpolating the two objectives provides further benefit. Every trained row is run with three seeds $\{42, 43, 44\}$; the seed controls LoRA initialization, data-shuffling order, and, for RL, rollout sampling. The standalone Bradley–Terry reward-model baselines are trained with all three seeds. For the subsequent $\tilde{R}$ policy-optimization pipeline, we use the reward model trained with seed 42 and hold it fixed across the three policy-training seeds. The reported $\tilde{R}$ policy variance therefore isolates policy-training and rollout randomness rather than reward-model initialization. The base model is Qwen3-4B for both domains. Online-RL rollouts use group size $K$=4 and sampling temperature 0.7; all evaluations use greedy decoding (temperature 0).

## A.2 UltraFeedback Prompts and Aggregation

Four dimension judges (helpfulness, honesty, instruction-following, truthfulness) each see the same input (instruction + two responses) but score only their own dimension on a 1–5 scale, using the shared template:

```
You are an evaluator focused on the {DIM}
dimension only. Rate each response's {DIM}
on a 1-5 scale (5 = best).
Task instruction: {instruction}
Response A: {A}
Response B: {B}
```

At training time each judge is a MADPO/DPO agent whose implicit reward is $\beta \log \pi_\theta / \pi_{\text{ref}}$ on its dimension prompt. Two aggregators play distinct roles. During data preparation, an ordinary least-squares regression of each completion's overall score (1–10) on its four gold dimension ratings (fit over 7,588 completions) quantifies how the dimensions relate to the overall judgment: coefficients {helpfulness 0.185, honesty 0.348, instruction-following 0.312, truthfulness 0.339}, intercept 1.921, with coefficient of determination $R^2$=0.57—the four dimensions linearly explain about half the variance in the overall score. At evaluation, the reported accuracies instead use a logistic aggregator refit per method: for each trained method, a logistic regression over its four per-dimension implicit-reward differences is fitted on the training pairs (symmetrized, so the intercept is $\approx 0$) and applied to the 187 held-out pairs, scoring the predicted winner against the gold overall choice.

*Per-dimension orientation and the joint tuple.* Each judge's chosen/rejected pair is oriented by *its own* dimension rating, so orientations need not agree across judges: on 13% of dimension pairs the dimension-preferred response is not the overall winner (e.g. the more honest response is the overall-worse one). The joint preference tuple is therefore constructed *factorwise*—$\mathbf{y}^w$ collects each judge's own dimension-chosen response and $\mathbf{y}^l$ its dimension-rejected one—so the single label $\mathbf{y}^w \succ \mathbf{y}^l$ holds by construction, one factor per agent, without requiring a coherent overall winner. Equivalently, the joint margin $\sum_i m_i$ compares the per-dimension-best assembly against the per-dimension-worst assembly rather than two literal responses. This construction differs from WebShop in an important respect. In WebShop, the per-agent orientations agree with the team-preferred trajectory, whereas UltraFeedback orients each factor independently according to its own dimension. The resulting training tuples satisfy the component-wise orientation used by Proposition 2 in the main paper, explaining why joint and independent training remain comparable in aggregate. At evaluation, however, the team prediction is judged against the overall human preference, which may disagree with one or more dimension-level ratings. This distinction motivates the consistent–conflict analysis in the main paper and separates interaction dependence from local–global preference compatibility.

## A.3 WebShop Prompts and Protocol

The search agent maps an instruction to a keyword query; the click agent maps the (instruction, titled results page) to a purchase plan:

**Search Agent**

```
You are shopping online. Given the
instruction, write a short search query
(keywords only).
Instruction: {instruction}
```

**Click Agent**

```
You are shopping online to satisfy:
{instruction}
Search results (ID | title | price): {page}
List items to click: product ID, then each
option, then "buy now". One per line.
```

**Protocol.** A rollout issues one search, reconstructs the titled results page (`ASIN | title≤80 chars | price`), then executes the plan line by line against the environment; if the plan does not purchase, a final "buy now" is attempted (identical across all methods). Chosen sides come from human demonstrations with mean environment reward 0.95 (filtered to reward $\geq$ 0.7); rejected sides are Qwen3-4B samples on the same prompt at temperature 0.9, seeded for reproducibility.

# B Additional Results

Figure 3 shows the per-update margin dynamics behind the aggregate results reported in the main paper.

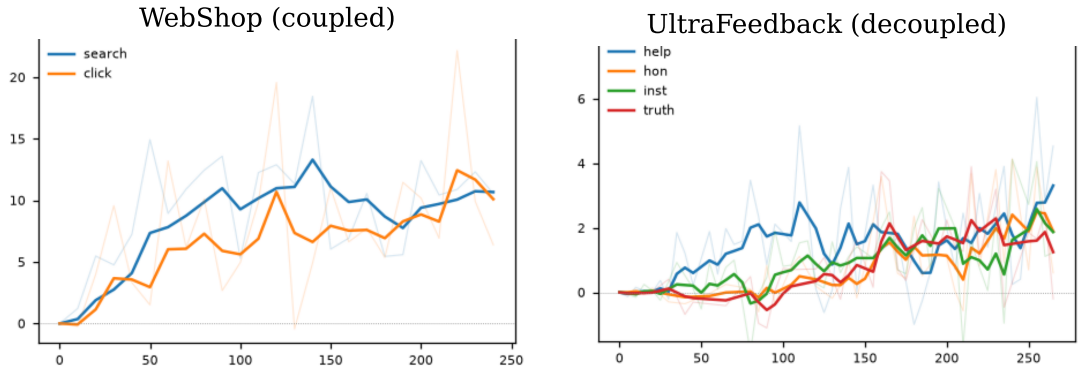


**Figure 3: MADPO training dynamics under the joint objective ($\alpha$=1), showing per-agent implicit margin $m_i$ ($y$) against training updates ($x$) for a representative seed. *Left:* on WebShop the two coupled agents (search, click) rise together under the shared team signal. *Right:* on UltraFeedback all four dimension judges (helpfulness, honesty, instruction-following, truthfulness) increase in concert even though the judges are interaction-independent. Faint curves are raw logged values; solid curves are EMA-smoothed.**

## B.1 Per-Seed Results

Tables 5 and 6 give the individual seed values behind the three-seed means in the main paper and expose the two failure modes discussed in Sec. 6: $\tilde{R}$ reward hacking and high independent-learner variance.

| WebShop row (300-goal) | s42 | s43 | s44 | mean |
|---|---|---|---|---|
| MADPO($\alpha$=1) (reward) | 56.0 | 54.0 | 59.0 | 56.3±2.5 |
| MADPO($\alpha$=0) (reward) | 53.0 | 49.0 | 55.7 | 52.5±3.4 |
| Single-model (reward) | 49.4 | 50.8 | 58.9 | 53.0±5.1 |
| MAGRPO $R^*$ (reward) | 74.0 | 73.6 | 73.9 | 73.9±0.2 |
| MAGRPO $R^*$ (succ.) | 42.3 | 43.3 | 41.0 | 42.2±1.2 |
| GRPO-indep (reward) | 57.7 | 55.3 | 72.2 | 61.7±9.1 |
| GRPO-indep (succ.) | 4.3 | 3.0 | 43.0 | 16.8±22.7 |
| MAGRPO $\tilde{R}$ (reward) | 52.2 | 51.2 | 51.7 | 51.7±0.5 |

**Table 5: WebShop per-seed values (%) on the 300-goal test set.**

| UltraFeedback (accuracy) | s42 | s43 | s44 | mean |
|---|---|---|---|---|
| MADPO($\alpha$=1) | 86.6 | 84.5 | 85.0 | 85.4±1.1 |
| MADPO($\alpha$=0) | 84.5 | 85.0 | 85.0 | 84.9±0.3 |
| Single-judge DPO | 85.6 | 83.4 | 86.1 | 85.0±1.4 |
| BT RM (single head) | 80.7 | 81.8 | 84.5 | 82.4±1.6 |
| BT RM (multi head) | 80.7 | 80.7 | 80.7 | 80.7±0.0 |
| MAGRPO $\tilde{R}$ | 54.0 | 58.8 | 61.5 | 58.1±3.8 |
| GRPO $\tilde{R}$ | 57.2 | 57.8 | 66.8 | 60.6±5.4 |

**Table 6: UltraFeedback per-seed accuracy (%).**

## B.2 Cost Accounting

Inference-time coordination trades additional tokens and latency for communication. WebShop already requires a two-stage search–click execution: the click agent acts after observing the result page generated by the search agent. The discussion baseline adds a further query-revision round after the initial result and plan, approximately doubling the decoded tokens per query. Preference-trained methods retain the environment's standard two-stage execution without introducing additional rounds. Their primary cost is one-time offline training, whereas the online-RL rows on WebShop additionally require 1,944 sampled environment episodes (486 goals, group size $K$=4) and the $\tilde{R}$ pipeline a preceding reward-model fit. All wall-clock figures are measured on one NVIDIA A40 GPU.

## B.3 WebShop Format-Mismatch Study

An earlier WebShop pair set (v1) presented the click agent with opaque product IDs, whereas the environment page the agent faces at evaluation lists titles and prices. Training and evaluating under this train/eval format mismatch suppressed the click agent's ability to discriminate products, and the human ≈62% first-result bias dominated; enriching the click prompt with titles and prices (v2, used throughout the main paper) restored a learnable click decision. We note this because it is the single largest protocol lesson in our study: preference-training pairs must be rendered in the same surface format the policy sees at deployment.

# C Implementation Pitfalls

In our implementation all agents share one frozen base model and differ only in LoRA adapters; the reference policy is the same base model with adapters disabled. This multi-adapter setting is memory-efficient but introduces failure modes that are easy to miss because training proceeds without any error. We document them here.

## C.1 Reference Forwards Must Precede Retained Policy Forwards

Adapter libraries expose the reference policy by temporarily disabling adapters (e.g., `disable_adapter()` in `peft`). Toggling adapter state mutates the wrapped modules in place, which invalidates autograd graphs built *before* the toggle. In a naive $N$-agent loop that interleaves each agent's policy forward with its reference forward, every policy graph except the last is built before some subsequent toggle: the backward pass then silently returns exactly zero gradient for agents $1, \ldots, N-1$, and training "succeeds" while $N-1$ adapters remain at initialization. Two cheap detectors catch this: (i) log per-agent gradient norms—affected agents sit at exactly zero from step one; (ii) inspect saved adapters—because `lora_B` matrices are zero-initialized, untrained agents ship adapters whose `lora_B` norms are identically zero. The fix is structural, reflected in Algorithm 1 (line 5) of the main paper: compute *all* reference log-probabilities first, under `no_grad` with adapters disabled, and only then run policy forwards whose graphs are retained for the backward step.

## C.2 Memory: Chunked Gather–LogSumExp

Sequence log-probabilities are usually computed from the full $(B, T, V)$ float32 logit tensor, which spikes peak memory for vocabularies around $10^5$. We never materialize the full log-softmax: logits are processed in chunks along the sequence dimension, and for each chunk we gather the target-token logit and subtract a vocabulary-wise logsumexp, accumulating per-sequence sums. Results are unchanged; peak activation memory drops by roughly the chunking ratio.

## C.3 Per-Agent Immediate Backward via a First-Order-Exact Surrogate

Backpropagating $\mathcal{L}_{\text{MADPO}}$ directly retains all $N$ agents' activation graphs simultaneously. We instead (i) compute all margins $m_j$ without gradients; (ii) form the detached scalar coefficients

$$c_i \;=\; \alpha\,\sigma\Big(-\textstyle\sum_j m_j\Big) \;+\; (1-\alpha)\,\sigma(-m_i);$$

(iii) for each agent $i$ in turn, recompute $m_i$ with gradients enabled and immediately backpropagate the surrogate $\ell_i = -\,c_i\,m_i$, freeing agent $i$'s graph before agent $i$+1's forward. By the shared-credit gradient and linearity of the $\alpha$-mix,

$$\begin{aligned}\nabla_{\theta_i}\mathcal{L}_{\text{MADPO}} &= -\Big[\alpha\,\sigma(-\textstyle\sum_j m_j) + (1-\alpha)\,\sigma(-m_i)\Big]\,\nabla_{\theta_i} m_i \\ &= \nabla_{\theta_i}\ell_i,\end{aligned}$$

since $c_i$ equals the bracketed coefficient evaluated at the same parameter values. The surrogate is therefore *exact* for the gradient (not an approximation); the cost is one extra no-grad forward per agent, and peak memory is bounded by a single agent's activation graph.

## C.4 Shell Pipelines Mask Training Failures

A final, mundane pitfall: launching training as `python train.py | tee log.txt` under `set -e` does not abort on failure, because the pipeline's exit status is `tee`'s. Multi-stage sweeps then keep running after a crashed stage, silently producing stale checkpoints. Use `set -o pipefail` (or write logs from within the trainer).