FACTOR-OPD: ALLOCATE REWARD-SPECIFIC CREDIT BEFORE COMBINING POLICY GRADIENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-reward reinforcement learning must balance objectives that can require different updates within the same response. Sequence-level reward aggregation leaves this token-credit problem unresolved, while teacher feedback from uncontrolled references can mix the effects of several reward properties. We introduce FACTOR-OPD, which allocates reward-specific token credit before combining objective gradients. Verifier-checked factorial references contrast high and low levels of each reward property while balancing the other verified factors. A frozen teacher scores identical student-sampled tokens under these references, and constrained allocation converts the resulting contrasts into positive, bounded, mean-one token weights. This preserves each component advantage's sign and response-wise mean. Parameter-space MGDA then combines the induced objective gradients, accounting for their geometry in the shared model parameters. Across six benchmarks spanning mathematical reasoning, code generation, and instruction following, FACTOR-OPD outperforms GRPO, GDPO, and GD^{2}PO with both Qwen3-4B and Qwen3-8B. Domain-average accuracy gains over GD^{2}PO range from 1.73 to 1.84 percentage points. Ablations support complementary benefits from token allocation and gradient coordination, including the contribution of token alignment. Reward-wise analyses further show higher correctness without reduced format compliance on Math and Code.

1 INTRODUCTION

Multi-reward reinforcement learning trains language models to satisfy several requirements within a single response (Liu et al., 2026b;a). A mathematical answer can be correct yet violate the required format, receiving positive correctness and negative format advantages relative to other sampled responses. Both judgments concern the same output, but its answer tokens and formatting delimiters need not receive identical feedback. Balancing response-level rewards alone does not determine where each objective's feedback should be applied. We study *how to allocate reward-specific credit within a response before combining objective gradients*. Figure 1 illustrates this distinction.

Existing methods address reward aggregation and conflict at different levels. GRPO compares responses within each prompt group (Shao et al., 2024); GDPO normalizes reward components separately before aggregation (Liu et al., 2026b); and GD^{2}PO uses conflict-aware rollout filtering and query-level reweighting (Liu et al., 2026a). These operations govern response-level feedback. Blockwise advantage estimation assigns objectives to predefined text segments (Pavlenko et al., 2026), while teacher-assisted methods provide dense feedback through privileged context, textual feedback, or critiques (Zhao et al., 2026; Hübotter et al., 2026; Xie et al., 2025; Wang et al., 2026). The remaining challenge is to make token feedback reward-specific. A correctness-positive hint may also be shorter and better formatted than its negative counterpart, so their likelihood contrast mixes several changing properties. A finer-grained signal alone does not separate these factors.

Our insight is to control the reference comparison before estimating reward-specific token credit. FACTOR-OPD builds a verifier-checked factorial panel that crosses high and low values of the target hint properties while keeping the task and instruction fixed. A frozen teacher scores the *same student-sampled tokens* under each accepted hint. Balanced contrasts measure the change in teacher log-probability associated with one hint factor, averaged over the other verified factors. Reference

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

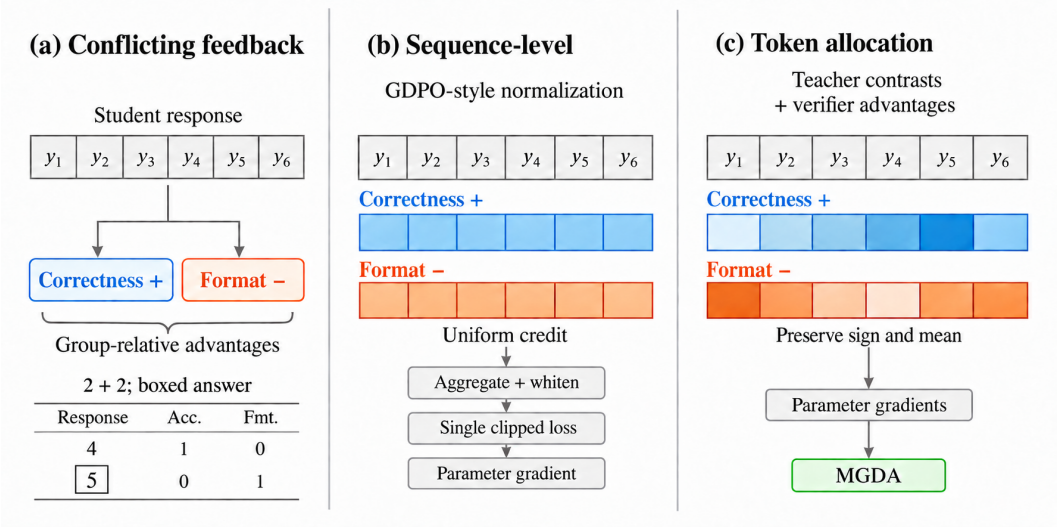


Figure 1: **Allocate token credit before combining gradients.** (a) Correctness and format rewards induce opposing group-relative advantages. (b) GDPO broadcasts component advantages across tokens and aggregates them into one policy loss. (c) FACTOR-OPD uses controlled teacher contrasts to allocate token credit, preserving each component advantage's sign and response-wise mean, then combines parameter gradients with MGDA. The example and credit shading are illustrative.

construction thus becomes part of the credit estimator: each contrast describes the teacher's token preference under a controlled change in measured hint properties.

These contrasts guide credit allocation while verifier outcomes retain control of each component's judgment. Positive, bounded token weights with response-wise mean one redistribute the component advantage without changing its sign or mean. Each allocated component then defines a separate policy loss. Because tokens share model parameters, similarity between token-credit vectors does not imply compatibility between the resulting parameter gradients. We therefore use the multiple-gradient descent algorithm (MGDA) (Sener & Koltun, 2018; Li et al., 2025) to combine those gradients. The verifier sets the outcome advantage, the teacher guides its allocation, and parameter-space MGDA selects the gradient mixture. Allocation precedes combination, so the solver acts on the directions induced by reward-specific credit.

We evaluate FACTOR-OPD on six benchmarks across mathematical reasoning, code generation, and instruction following with Qwen3-4B and Qwen3-8B (Yang et al., 2025). It achieves the highest accuracy in all twelve model–benchmark comparisons, improving domain-average accuracy over GD^{2}PO by 1.73–1.84 percentage points. Ablations show complementary gains from token allocation and parameter MGDA, while reference and token-permutation controls support the importance of reward-specific, aligned feedback. Tool-calling experiments extend the comparison to correctness and length rewards; reward-wise analyses on Math and Code show higher correctness without lower format compliance. In summary, our contributions are as follows:

- We introduce verifier-checked factorial contrasts for reward-specific token feedback and a bounded, mean-one allocation that preserves each component advantage's sign and response-wise mean.
- We combine the allocated objectives using parameter-space MGDA and characterize why token-credit geometry cannot generally substitute for parameter-gradient geometry.
- We demonstrate benchmark gains across three domains and two model scales, with ablations and reward-wise analyses supporting the complementary roles of credit allocation and gradient combination.

2 Related Work

Multi-reward policy optimization. RLHF aligns language models with judgments of response quality (Ouyang et al., 2022), while multi-reward methods explicitly balance distinct objectives. MODPO uses weighted preference objectives (Zhou et al., 2024), Safe RLHF constrains helpfulness optimization by safety cost (Dai et al., 2024), and dynamic weighting adapts reward trade-offs during training (Lu et al., 2026). Within PPO-based group-relative optimization (Schulman et al., 2017; Shao et al., 2024), MO-GRPO uses variance-based reward normalization (Ichihara et al., 2026), whereas GDPO normalizes component advantages before aggregation (Liu et al., 2026b). DAPO filters groups with uniform correctness (Yu et al., 2025), and GD^{2}PO filters rollouts with conflicting component advantages and reweights queries by the retained rollout fraction (Liu et al., 2026a). Gradient-based methods instead coordinate objective-specific update directions. MGDA selects a minimum-norm convex gradient combination in multi-task learning and multi-objective RL (Sener & Koltun, 2018; Zhan & Cao, 2021). PCGrad projects conflicting gradients (Yu et al., 2020), CAGrad uses worst local task improvement to regularize average-loss descent (Liu et al., 2021), and GAPO applies multiple-gradient descent to language model alignment (Li et al., 2025). FACTOR-OPD applies parameter-space MGDA after reward-specific token allocation, coordinating gradients whose directions already reflect where each component advantage acts within a response.

Fine-grained credit assignment and distillation. Process supervision evaluates intermediate reasoning steps (Lightman et al., 2024), while blockwise advantage estimation assigns each objective's credit to a corresponding text segment (Pavlenko et al., 2026). On-policy distillation provides dense teacher feedback: GKD learns from teacher predictions on student-generated sequences (Agarwal et al., 2024). OPSD conditions a self-teacher on privileged solutions (Zhao et al., 2026), and SDPO uses textual feedback or successful rollouts for self-distillation (Hübotter et al., 2026). CAPO localizes credit through generated critiques (Xie et al., 2025), and CrEST preserves verifier-derived advantage signs while modulating token magnitudes with privileged teacher feedback (Wang et al., 2026). FACTOR-OPD uses a factorial reference panel to separate contrasts across measured reward properties on the same sampled tokens. These contrasts redistribute component advantages while preserving their signs and response-wise means, rather than directly defining a teacher–student divergence loss or requiring predefined objective blocks.

3 Preliminaries

Problem Setup. For prompts $x \sim \mathcal{D}$, a student policy π_θ generates responses evaluated by K reward functions $r_k(x, y)$. Multi-reward optimization concerns the vector $(J_1(\theta), \ldots, J_K(\theta))$, where

$$J_k(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[r_k(x, y)], \qquad k = 1, \ldots, K. \tag{1}$$

Outcome-supervised GRPO samples G responses per prompt from $\pi_{\theta_{\text{old}}}$ and normalizes their scalar rewards within each group to obtain advantages A_i (Shao et al., 2024). Define the token probability ratio $\rho_{i,t}(\theta) = \pi_\theta(y_{i,t} \mid x_i, y_{i,<t})/\pi_{\theta_{\text{old}}}(y_{i,t} \mid x_i, y_{i,<t})$, where x_i is the prompt for response i. Over batches $\mathcal{B}$ of these prompt groups, the global-token-mean clipped policy-gradient (PG) surrogate is

$$\mathcal{L}_{\text{PG}}(\theta; A) = -\mathbb{E}_{\mathcal{B}}\Big[\frac{1}{M_{\mathcal{B}}}\sum_{i,t} m_{i,t}\, \min\{\rho_{i,t}A_i, \text{clip}(\rho_{i,t}, 1-\delta, 1+\delta)A_i\}\Big]. \tag{2}$$

Here $m_{i,t}$ marks valid response tokens, $T_i = \sum_t m_{i,t} > 0$, $M_{\mathcal{B}} = \sum_i T_i$, and δ is the clipping threshold (Appendix B). This surrogate uses old-policy samples, whereas J_k evaluates the current policy's expected reward. Equation 2 isolates the PG term with token-mean reduction; original GRPO uses response-wise averaging and adds KL regularization (Shao et al., 2024). For multiple rewards, let $r_{i,k} = r_k(x_i, y_i)$. Reward-wise group normalization, as used in GDPO (Liu et al., 2026b), gives

$$a_{i,k} = \frac{r_{i,k} - \mu_{x_i,k}}{\sigma_{x_i,k} + \epsilon}. \tag{3}$$

Here $\mu_{x_i,k}$ and $\sigma_{x_i,k}$ are the within-prompt mean and standard deviation, with $\epsilon > 0$ for stability. Zero-variance components receive zero advantage.

Why token-level credit? Equation 2 already sums token-level losses, but broadcasts the same A_i across each response. With separate reward components, uniform credit assigns every valid token

$$b_{i,k,t}^{\text{uniform}} = a_{i,k}. \tag{4}$$

Here $b_{i,k,t}$ is the component's coefficient before clipping. Shared coefficients do not imply identical token gradients, since token probabilities depend on different prefixes. Correctness and format advantages can have opposite signs yet act on the same tokens; changing their relative weights leaves this within-response allocation uniform.

This motivates *reward-specific token credit assignment*: replace $a_{i,k}$ by $a_{i,k}w_{i,k,t}$ while preserving the verifier's component-level judgment. The challenge is to obtain feedback that separates reward factors and allocate it within each response. Section 4 constructs this allocation, then combines the resulting parameter gradients with MGDA.

4 METHOD

FACTOR-OPD constructs controlled teacher contrasts, converts them into constrained token weights, and combines the induced parameter gradients with MGDA. The verifier supplies the component advantage throughout; privileged hints are used only by the frozen teacher, never by the student. Figure 2 summarizes the training pipeline from reward-specific teacher feedback to the combined parameter update.

4.1 REWARD-SPECIFIC TOKEN FEEDBACK

For K reward components, a complete factorial panel contains 2^K references $h_x(z)$, with $z \in \{-1, +1\}^K$ encoding verified low and high factor levels. For any component k, let $z^{k,-}$ and $z^{k,+}$ differ only in their kth coordinate, with all remaining coordinates z_{-k} fixed (Table 1). Each pair changes the target factor while preserving non-target reward values. All references use a common neutral instruction and exclude the target response itself. Panels without a complete, verified crossing receive uniform credit; construction details and a two-factor example appear in Appendix B.

Table 1: Paired references for any reward k, holding all other factors fixed.

Factor k	Other factors	Reference
Low (-1)	z_{-k}	$h_x(z^{k,-})$
High $(+1)$	z_{-k}	$h_x(z^{k,+})$

A frozen initial-policy teacher π_{ref} scores the same student-sampled tokens and prefixes under every reference:

$$\begin{aligned} \ell_{i,t}(z) &= \log \pi_{\text{ref}}\big(y_{i,t} \mid x_i, h_{x_i}(z), y_{i,<t}\big), \\ \tau_{i,k,t} &= \frac{1}{2^{K-1}} \sum_{z_{-k}} \big[\ell_{i,t}(z^{k,+}) - \ell_{i,t}(z^{k,-})\big]. \end{aligned} \tag{5}$$

The sum averages the high-minus-low contrast over all 2^{K-1} settings of the other factors $z_{-k} \in \{-1, +1\}^{K-1}$. The complete design therefore requires 2^K teacher-scoring conditions per response, not just the pair illustrated in Table 1.

This contrast measures a main effect of the *constructed hint intervention* on teacher log-probability, not a causal token effect on task reward (Appendix A). On-policy distillation here scores student-sampled trajectories without generating replacement targets or adding a separate full-vocabulary KL objective.

4.2 CONSTRAINED TOKEN CREDIT ALLOCATION

We replace uniform credit (Equation 4) with $b_{i,k,t} = a_{i,k}w_{i,k,t}$. For $0 \le \gamma < 1$, require

$$\frac{1}{T_i} \sum_t m_{i,t} w_{i,k,t} = 1, \qquad 1 - \gamma \le w_{i,k,t} \le 1 + \gamma. \tag{6}$$

The positive, bounded, mean-one weights preserve each component's mean credit and sign. Advantages, teacher scores, and allocation weights are detached from the actor's computation graph.

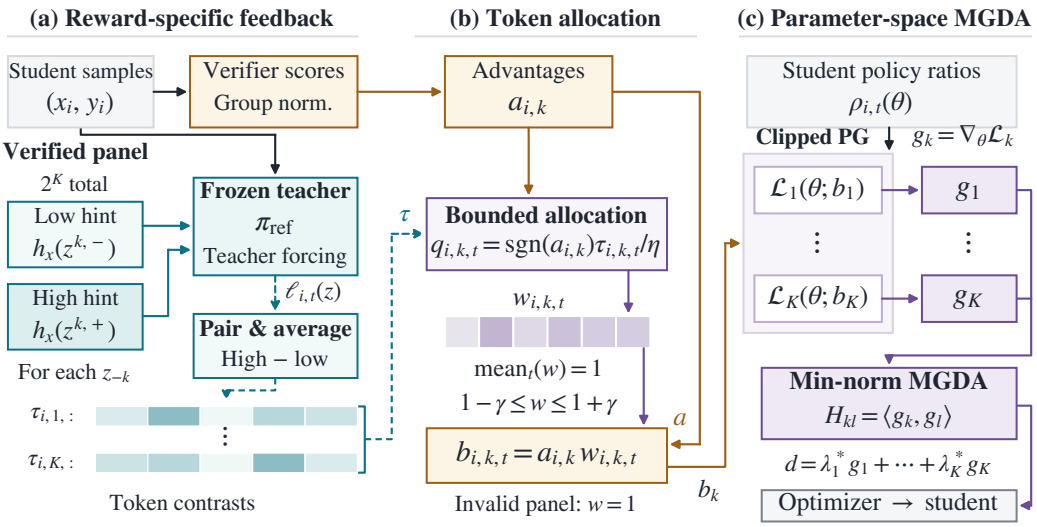


Figure 2: Overview of FACTOR-OPD for K reward components. (a) A frozen teacher scores identical student-sampled tokens under controlled reference pairs, averaging high–low contrasts over the other factors. (b) Contrasts and verifier advantages determine positive, bounded, mean-one weights, giving $b_{i,k,t} = a_{i,k} w_{i,k,t}$. (c) Separately clipped losses on the same minibatch yield parameter gradients, which MGDA combines over eligible objectives for the optimizer. Feedback and allocation are detached; cell shading is schematic.

Define $q_{i,k,t} = \text{sgn}(a_{i,k})\tau_{i,k,t}/\eta$, with temperature $\eta > 0$. Larger contrasts receive more positive credit when $a_{i,k} > 0$; smaller contrasts receive larger penalties when $a_{i,k} < 0$. For valid tokens, the entropy-regularized allocation takes the form

$$v^*_{i,k,t} = \text{clip}\big(e^{q_{i,k,t} - \nu_{i,k}},\, 1-\gamma,\, 1+\gamma\big)\,, \tag{7}$$

where the scalar $\nu_{i,k}$ enforces mean one over valid tokens. Appendix A gives the optimization problem, bisection solver, and equivalence to centered contrasts. A support weight $s_{i,k} \in [0,1]$, constant across valid tokens, interpolates with uniform credit:

$$w_{i,k,t} = 1 + s_{i,k}(v^*_{i,k,t} - 1). \tag{8}$$

Invalid panels use $s_{i,k} = 0$; padding receives zero credit.

Proposition 1 (Component-credit invariants) *For a valid response, the construction above satisfies $T_i^{-1}\sum_t m_{i,t} b_{i,k,t} = a_{i,k}$ and $|b_{i,k,t} - a_{i,k}| \le \gamma |a_{i,k}|$. If $a_{i,k} \neq 0$, its sign is preserved at every valid token. If $\gamma = 0$ or $s_{i,k} = 0$, the component is exactly its uniform-credit counterpart.*

The sign is that of the group-relative advantage, not the raw reward. Zero advantages remain zero, including all-tied components. These invariants precede loss clipping and gradient combination; they do not preserve parameter gradients or guarantee task improvement. See Appendix A for the proof and Appendix B for support settings.

4.3 Parameter-Space MGDA

Each transported component b_k defines a separate policy loss $\mathcal{L}_k$ and parameter gradient g_k. Fixed scalarization uses prescribed weights to combine these gradients. MGDA instead selects a minimum-norm convex combination (Sener & Koltun, 2018). Choosing token weights and choosing gradient weights are distinct operations: the first changes each objective's direction, while the second combines those directions into one update. Let $\widehat{\mathcal{L}}_{\text{PG}}$ denote the empirical version of Equation 2 on the sampled minibatch. Replacing A_i by $b_{i,k,t}$ gives

$$\mathcal{L}_k(\theta) = \widehat{\mathcal{L}}_{\text{PG}}(\theta; b_k), \qquad g_k = \nabla_\theta \mathcal{L}_k(\theta). \tag{9}$$

Each component is clipped separately on the same global minibatch, with the same token-mean normalization $M_{\mathcal{B}}$ and no combined-advantage whitening. Appendix A gives the expanded loss. This section specifies raw parameter MGDA; distributed execution, other normalizations, and variant reporting are detailed in Appendix B.

For eligible objectives $k, l \in \mathcal{A}$, form the parameter Gram matrix $H_{kl} = \langle g_k, g_l \rangle$. Let $\Delta_{\mathcal{A}}$ be the simplex of nonnegative weights on $\mathcal{A}$ that sum to one. MGDA solves

$$\lambda^* = \arg\min_{\lambda \in \Delta_{\mathcal{A}}} \lambda^\top H \lambda, \qquad d = \sum_{k \in \mathcal{A}} \lambda_k^* g_k. \tag{10}$$

Objectives without eligible samples are excluded from $\mathcal{A}$. The detached coefficients are shared across the global minibatch, and d is passed to the optimizer.

At differentiable points, the exact solution satisfies $g_k^\top d \geq \|d\|_2^2$ for every $k \in \mathcal{A}$. Thus $-d$ is a common first-order descent direction when $d \neq 0$. This property concerns sampled surrogate losses before optimizer preconditioning, not finite-step task-reward improvements (Appendix A).

Credit-space proxies omit the policy's shared-parameter geometry and can yield different mixing weights (Appendix A). Removing token allocation gives outcome-only MGDA, while GDPO also changes the aggregation, whitening, and clipping path. Our matched controls therefore hold the per-objective loss path fixed while varying allocation and gradient combination (Appendix D).

5 Experiments

We evaluate FACTOR-OPD on mathematical reasoning, code generation, and instruction following across two model scales. Our experiments examine whether allocating reward-specific token credit before combining objective gradients improves benchmark accuracy over existing multi-reward optimization methods.

5.1 Experimental Setup

Tasks and models. The main comparison uses Qwen3-4B and Qwen3-8B (Yang et al., 2025) with thinking disabled. For each model size, we train separate policies for mathematical reasoning, code generation, and instruction following. Training uses correctness and format rewards, with method-specific aggregation of the component scores. Training and evaluation configurations are given in Appendix C.

Benchmarks and metrics. Math uses AIME 2024 and AIME 2025. Code uses the selected LiveCodeBench v5 and v6 sets (Jain et al., 2025). IF uses the single-turn IFEval and IFBench sets with strict prompt-level accuracy (Zhou et al., 2023; Pyatkin et al., 2025). The main table reports mean per-response correctness: avg@16 for Math, avg@8 for Code, and one sampled response per prompt for IF (avg@1). Scores are averaged over repeated evaluations with matched decoding settings across all methods.

Baselines. We compare FACTOR-OPD against the pre-RL models (Base) and three RL baselines. GRPO (Shao et al., 2024) estimates advantages by normalizing rewards within each prompt group. GDPO (Liu et al., 2026b) normalizes each reward component separately before aggregation. GD^{2}PO (Liu et al., 2026a) combines conflict-aware rollout filtering with query-level consensus reweighting.

5.2 Main Results

FACTOR-OPD leads all twelve model–benchmark comparisons in Table 2, exceeding the strongest baseline in each pair by 1.23–2.23 percentage points. The gains also hold after averaging within domains: GD^{2}PO has the highest baseline averages in all six domain–model settings, yet FACTOR-OPD improves them by 1.73–1.84 points. Baseline rankings, by contrast, depend on the task. GDPO has higher Math averages than GRPO, whereas GRPO has higher IF averages at both model sizes. Even on Qwen3-4B AIME25 and LCB v5, where GRPO is the strongest baseline, FACTOR-OPD remains ahead.

Table 2: Three-domain benchmark accuracy (%); headers specify the number of sampled responses per problem in each evaluation. Scores are means over repeated evaluations. Avg is the unweighted mean across the two benchmarks in each domain. Each model block uses non-thinking, domain-specific policies. Base is the common initialization. IF columns use strict prompt-level correctness. Best results within each model block are bold.

Method	Math (avg@16)			Code (avg@8)			IF (avg@1)		
	AIME24	AIME25	Avg	LCB v5	LCB v6	Avg	IFEval	IFBench	Avg
Qwen3-4B									
Base	19.79	21.46	20.63	21.03	25.93	23.48	80.78	23.33	52.05
GRPO	20.63	21.67	21.15	23.73	26.43	25.08	81.52	26.00	53.76
GDPO	22.29	20.42	21.35	21.93	27.64	24.79	81.52	23.33	52.42
GD^{2}PO	22.41	21.15	21.78	22.74	27.89	25.32	83.17	27.67	55.42
FACTOR-OPD	**24.12**	**23.12**	**23.62**	**24.97**	**29.12**	**27.05**	**85.02**	**29.33**	**57.18**
Qwen3-8B									
Base	26.25	21.88	24.06	24.03	24.43	24.23	80.78	27.00	53.89
GRPO	26.88	21.88	24.38	24.03	25.86	24.94	85.77	27.33	56.55
GDPO	28.33	21.88	25.10	24.63	26.07	25.35	85.21	27.33	56.27
GD^{2}PO	28.45	22.61	25.53	25.44	26.32	25.88	86.86	31.67	59.27
FACTOR-OPD	**30.16**	**24.58**	**27.37**	**27.67**	**27.55**	**27.61**	**88.71**	**33.33**	**61.02**

Taken together, these results support reward-specific token allocation followed by parameter-gradient coordination as an effective multi-reward training strategy across the evaluated domains and model scales. The complete pipeline improves on both component-wise normalization and conflict-aware rollout selection. Section 5.3 examines the contributions of allocation and gradient combination separately.

5.3 Ablation Study

We organize the ablation study around two comparisons on Qwen3-4B using the domain-specific training and evaluation protocols in Section 5.1. Both report domain-average benchmark accuracy, averaged over three training seeds. Appendix D gives the seed settings and matched controls.

[Abl.1] Token Allocation and Gradient Combination. Table 3 crosses uniform versus factorial token allocation with fixed weights versus parameter MGDA. Uniform allocation broadcasts each component advantage across the response; factorial allocation redistributes it using the teacher contrasts. The fixed-weight controls combine separately computed component gradients. All four configurations retain the same per-component clipped losses, token-mean normalization, and optimizer. Comparing allocation rules under the same combiner isolates token allocation, while comparing combiners under the same allocation isolates MGDA. The crossed design also tests whether the two stages interact.

Table 3: Token allocation and gradient combination on Qwen3-4B. Domain-average accuracy (%), averaged over three seeds; best results are bold.

Token allocation	Gradient combination	Math	Code	IF
Uniform	Fixed weights	21.84	25.36	55.43
Factorial	Fixed weights	22.46	25.92	56.02
Uniform	Parameter MGDA	22.29	25.98	55.90
Factorial (FACTOR-OPD)	Parameter MGDA	**23.62**	**27.05**	**57.18**

FACTOR-OPD achieves the highest accuracy in all three domains. Removing factorial allocation reduces Math, Code, and IF accuracy by 1.33, 1.07, and 1.28 percentage points, respectively. Replacing parameter MGDA with fixed gradient weights reduces accuracy by 1.16, 1.13, and 1.16 points. Both stages also improve on the uniform-credit, fixed-weight control when applied individ-

ually: allocation gains 0.56–0.62 points, while MGDA gains 0.45–0.62 points. Thus, neither stage alone accounts for the full method's gains.

The benefit of factorial allocation is larger under MGDA than under fixed weights. The resulting allocation–combination interaction estimates (Appendix D) are positive in all three domains: 0.71 points on Math, 0.51 on Code, and 0.69 on IF. These point estimates suggest complementary contributions from reward-specific token allocation and parameter-gradient combination.

[Abl.2] Controlled References and Token Alignment. Holding parameter MGDA unchanged, Table 4 compares natural-sibling marginal contrasts (Marginal), controlled factorial contrasts, and factorial allocation with permuted token weights (Permuted). Comparing natural-sibling and factorial contrasts tests the role of controlled reference construction. The permutation control keeps the factorial references and teacher scores unchanged, then applies one shared permutation across reward components within each response's valid token positions. It preserves the weight distributions, bounds, and component budgets, while disrupting their alignment with the sampled tokens. This comparison tests whether token-specific alignment matters beyond nonuniform weighting. The teacher, hint-count and length budgets, allocation rule, and loss path are matched across the three configurations.

Table 4: Reference construction and token alignment on Qwen3-4B. All use parameter MGDA; metrics follow Table 3.

Feedback	Math	Code	IF
Marginal	22.71	26.21	56.24
Permuted	22.93	26.36	56.41
FACTOR-OPD	**23.62**	**27.05**	**57.18**

Aligned factorial credit achieves the highest accuracy in all three domains. It exceeds natural-sibling marginal contrasts by 0.91, 0.84, and 0.94 percentage points on Math, Code, and IF, respectively, supporting the controlled-reference design as a whole. Permuting the factorial token weights reduces accuracy by 0.69, 0.69, and 0.77 points while preserving their distributions and component budgets. This comparison supports an additional contribution from matching reward-specific credit to the sampled tokens, beyond nonuniform weighting alone.

The permuted control nevertheless exceeds natural-sibling contrasts by 0.15–0.22 points and uniform-credit MGDA (Table 3) by 0.38–0.64 points. Some benefit therefore persists after permutation, while preserving token-specific alignment yields the best observed accuracy.

6 Detailed Analysis

We examine generalization to tool calling under a different reward combination, followed by correctness–format trade-offs on Math and Code with Qwen3-4B. Appendix E gives the analysis protocols.

[Ana.1] Generalization to Tool Calling. We test generalization with the correctness–length tool-calling setting of GD^{2}PO (Liu et al., 2026a). Table 5 compares FACTOR-OPD with published baselines on Qwen2.5-1.5B-Instruct, Qwen2.5-3B-Instruct, and Llama-3.1-8B-Instruct. Training uses RLLA and evaluation uses API-Bank (Li et al., 2023); the length reward favors longer reasoning traces.

The reconstructed Correct Acc. exceeds the strongest published baseline by approximately 1.10, 1.04, and 1.12 percentage points, respectively. The reported length reward matches the best displayed mean of 1.00, yielding reconstructed Overall gains of 0.017, 0.010, and 0.011. These point estimates suggest generalization beyond the main correctness–format setting.

FACTOR-OPD leads Level 1 and Level 2 on every backbone and all three levels on Qwen2.5-3B-Instruct. However, Level 3 accuracy remains 2.88 points below GRPO on Qwen2.5-1.5B-Instruct and 1.43 points below GD^{2}PO-SNR on Llama-3.1-8B-Instruct. The aggregate gain therefore does not imply a uniform advantage on the hardest level.

[Ana.2] Reward-Wise Trade-offs. Benchmark accuracy does not reveal whether task improvements accompany a change in format compliance. We therefore compare FACTOR-OPD, GRPO, GDPO, and GD^{2}PO using the Math and Code correctness and format rewards on held-out data after full training. Table 6 reports the two components separately for Qwen3-4B.

Table 5: Tool calling with correctness and length rewards. Baseline results are from Table 1 of Liu et al. (2026a) (five-run means, with standard deviations where reported). Accuracy is in %; Length Rew. is in $[0, 1]$, and Overall = Correct Acc./100 + Length Rew. Bold marks the best aggregate scores within each backbone; second-best Correct Acc. and Overall are underlined. For FACTOR-OPD, Correct Acc. standard deviations use zero-covariance propagation (Appendix E).

Model	Method	Level 1 Acc.	Level 2 Acc.	Level 3 Acc.	Correct Acc.	Length Rew.	Overall
Qwen2.5-1.5B-Instruct	GRPO	56.59±4.37	43.28±5.28	40.00±1.49	51.46±3.24	0.99±0.01	1.505
	GDPO	56.69±6.88	44.78±8.04	36.33±5.02	50.89±6.37	**1.00**±0.00	1.509
	GD^{2}PO-Hard	59.95±4.04	44.48±6.28	36.03±7.25	<u>52.96</u>±3.95	0.99±0.01	1.520
	GD^{2}PO-SNR	58.59±3.47	45.97±8.34	36.64±3.54	52.36±3.16	**1.00**±0.00	<u>1.524</u>
	FACTOR-OPD	60.78±2.48	47.14±3.22	37.12±4.52	**54.06**±1.97	**1.00**±0.00	**1.541**
Qwen2.5-3B-Instruct	GRPO	67.22±0.96	56.42±3.87	45.50±2.20	61.24±1.02	0.99±0.01	1.602
	GDPO	67.42±2.13	56.12±7.13	46.57±4.00	61.57±1.96	**1.00**±0.00	1.616
	GD^{2}PO-Hard	69.27±1.33	57.61±3.89	46.11±2.94	<u>62.88</u>±1.56	**1.00**±0.00	<u>1.629</u>
	GD^{2}PO-SNR	68.57±2.28	58.21±3.81	45.49±2.68	62.35±1.99	**1.00**±0.00	1.624
	FACTOR-OPD	70.12±1.56	59.38±3.34	47.35±2.91	**63.92**±1.28	**1.00**±0.00	**1.639**
Llama-3.1-8B-Instruct	GRPO	63.91±3.45	44.78±7.76	42.44±4.91	57.05±3.06	**1.00**±0.00	1.571
	GDPO	60.95±2.13	42.69±5.84	45.19±3.68	55.44±2.01	**1.00**±0.00	1.554
	GD^{2}PO-Hard	64.81±5.74	43.88±7.21	48.09±6.66	<u>58.79</u>±4.95	**1.00**±0.00	<u>1.588</u>
	GD^{2}PO-SNR	64.11±7.12	42.39±12.24	51.00±5.02	<u>58.79</u>±5.05	**1.00**±0.00	<u>1.588</u>
	FACTOR-OPD	65.68±4.02	45.74±5.88	49.57±4.63	**59.91**±2.95	**1.00**±0.00	**1.599**

On Math, FACTOR-OPD attains the highest correctness (55.78%) and format compliance (91.52%). It improves over GD^{2}PO by 0.84 and 3.00 percentage points, respectively. The baseline ordering reveals an accuracy–format tradeoff: GDPO improves format compliance over GRPO by 6.25 points but lowers correctness by 1.75 points. FACTOR-OPD exceeds both methods on both rewards.

Table 6: Held-out Math and Code reward scores (%) after full training of Qwen3-4B. Bold marks the best score in each domain.

Domain	Method	Correct	Format
Math	GRPO	53.13	84.38
	GDPO	51.38	90.63
	GD^{2}PO	54.94	88.52
	FACTOR-OPD	**55.78**	**91.52**
Code	GRPO	46.88	96.88
	GDPO	43.75	**100.00**
	GD^{2}PO	47.24	**100.00**
	FACTOR-OPD	**47.97**	**100.00**

On Code, format compliance reaches 100% for GDPO, GD^{2}PO, and FACTOR-OPD, so the remaining distinction is correctness. FACTOR-OPD achieves 47.97% correctness, exceeding GD^{2}PO by 0.73 points and GDPO by 4.22 points at the same format score. Across both domains, the reported component means show higher correctness with no reduction in format compliance relative to any baseline.

7 Conclusion

We introduced FACTOR-OPD, which combines reward-specific token credit assignment with parameter-space MGDA. Verifier-checked factorial references provide controlled teacher contrasts for token allocation that preserves each component advantage's sign and response-wise mean. Across mathematical reasoning, code generation, and instruction following, FACTOR-OPD improves benchmark accuracy over GRPO, GDPO, and GD^{2}PO on Qwen3-4B and Qwen3-8B. Ablations support complementary benefits from token allocation and gradient coordination, including the importance of token alignment. Reward-wise analyses show higher correctness without lower format compliance on Math and Code. These findings highlight the value of allocating reward credit before combining objective gradients.

AI USE STATEMENT

Generative AI tools were used for language polishing and assistance with figure preparation. All technical content, experiments, and conclusions were developed and verified by the authors.

ETHICS STATEMENT

This work studies reward-specific credit assignment using existing benchmarks for mathematical reasoning, code generation, instruction following, and tool calling. The method optimizes specified reward functions, whose coverage depends on the underlying verifiers and data. Benchmark improvements do not guarantee factual reliability, code security, or safe tool use. Applications should use isolated code execution, restricted tool permissions, and domain-specific validation, while respecting the licenses and privacy constraints of the underlying data and models.

REPRODUCIBILITY STATEMENT

Appendix A provides the derivations for factorial contrasts, bounded token allocation, and the local MGDA property. Appendix B describes reference construction, token masks, training updates, and distributed gradient combination. Reward definitions, training and evaluation settings, and checkpoint selection are given in Appendix C. Appendix D specifies the matched ablation controls and seed aggregation. Appendix E details the tool-calling comparison, aggregate-score calculations, and held-out reward-component evaluation. Appendix F summarizes implementation checks and the scope of the computational evidence.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Juntao Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe RLHF: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations*, volume 2024, pp. 50750–50777, 2024.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. MO-GRPO: Mitigating reward hacking of group relative policy optimization on multi-objective problems. *Transactions of the Association for Computational Linguistics*, 14:1867–1882, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida I. Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Chengao Li, Hanyu Zhang, Yunkun Xu, Hongyan Xue, Xiang Ao, and Qing He. Gradient-adaptive policy optimization: Towards multi-objective alignment of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11214–11232, 2025.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 3102–3116, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in neural information processing systems*, 34:18878–18890, 2021.

Haotian Liu, Yihao Liu, Jingwei Ni, Siyuan Huang, Xinpeng Liu, Pengyu Cheng, Jiajun Song, Ruijin Ding, Junfeng Li, Zhechao Yu, et al. GD^{2}PO: Mitigating multi-reward conflicts via group-dynamic reward-decoupled policy optimization. *arXiv preprint arXiv:2606.16771*, 2026a.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, et al. GDPO: Group reward-decoupled normalization policy optimization for multi-reward RL optimization. *arXiv preprint arXiv:2601.05242*, 2026b.

Yining Lu, Zilong Wang, Shiyang Li, Xin Liu, Changlong Yu, Qingyu Yin, Zhan Shi, Zixuan Zhang, and Meng Jiang. Learning to optimize multi-objective alignment through dynamic reward weighting. *Transactions of the Association for Computational Linguistics*, 14:1051–1073, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Kirill Pavlenko, Alexander Golubev, Simon Karasik, and Boris Yangel. Blockwise advantage estimation for multi-objective RL with verifiable rewards. *arXiv preprint arXiv:2602.10231*, 2026.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2025.

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

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zechuan Wang, Siyuan Lu, Hongxuan Zhang, Linjian Mo, Chenyi Zhuang, and Leilei Gan. Teach the magnitude, not the direction: Verifier-bounded credit assignment for multi-turn multi-step LLM agents. *arXiv preprint arXiv:2608.13179*, 2026.

Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. CAPO: Towards enhancing LLM reasoning through generative credit assignment. *arXiv preprint arXiv:2508.02298*, 2025.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2025.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Huixin Zhan and Yongcan Cao. Efficient multi-objective reinforcement learning via multiple-gradient descent with iteratively discovered weight-vector sets. *Journal of Artificial Intelligence Research*, 70:319–349, 2021.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10586–10613, 2024.

A Algebraic Properties and Their Scope

Full-factorial contrast. For $C = 2^K$ cells, exactly $C/2$ rows have $z_k = +1$ and $C/2$ have $z_k = -1$. Thus $(2/C)\sum_z z_k \ell(z)$ is the mean over positive cells minus the mean over negative cells. Pairing cells with the same z_{-k} gives the high-minus-low average in Equation 5. Let Z contain the complete set of factor vectors z as rows. Its columns are balanced and orthogonal, with $Z^\top Z = 2^K I$. The contrast therefore requires no regression on potentially collinear naturally observed reward vectors. Expanding a response over the complete Boolean basis, $\ell(z) = \sum_{S\subseteq\{1,\dots,K\}} \beta_S \prod_{j\in S} z_j$, gives $\tau_k = 2\beta_{\{k\}}$ by orthogonality. Higher-order terms are averaged out in this particular main-effect estimand; their substantive importance is not denied. A balanced orthogonal array only ensures pairwise column orthogonality. If higher-order terms alias its main-effect columns, the same contrast cannot be interpreted as the full-factorial effect without further assumptions.

Transport solution. Equation 7 solves the entropy-regularized allocation

$$v^*_{i,k} = \arg\max_{v} \sum_{t:m_{i,t}=1} \big(q_{i,k,t} v_t - v_t \log v_t\big) \quad \text{subject to } T_i^{-1} \sum_{t:m_{i,t}=1} v_t = 1, \qquad 1-\gamma \le v_t \le 1+\gamma, \qquad 0 \le \gamma < 1. \tag{11}$$

For $\gamma > 0$, the objective in Equation 11 is strictly concave over a nonempty convex feasible set. For a token not on its bound, stationarity yields $q_t - \log v_t - 1 - \xi = 0$. Absorbing $1 + \xi$ into ν gives $v_t = \exp(q_t - \nu)$. The bound constraints clip this value:

$$v_t = \mathrm{clip}\big(\exp(q_t - \nu), 1-\gamma, 1+\gamma\big).$$

Its sum is continuous and non-increasing in ν, permitting bisection; the optimal weights are unique even when a dual multiplier is not. For $\gamma = 0$, the feasible set consists only of the all-one vector. Adding a constant to every q_t changes the objective by a constant on the feasible set, so it does not change the weights. In particular, let $\bar\tau_{i,k} = T_i^{-1}\sum_t m_{i,t}\tau_{i,k,t}$ and $c_{i,k} = \mathrm{sgn}(a_{i,k})\bar\tau_{i,k}/\eta$. Centering the contrasts replaces $q_{i,k,t}$ by $q_{i,k,t} - c_{i,k}$. Replacing $\nu_{i,k}$ by $\nu_{i,k} - c_{i,k}$ then leaves Equation 7 unchanged. Thus centering may be used for numerical stability but is unnecessary in the mathematical definition of the allocation.

Proof of Proposition 1. The optimal weights have mean one over the valid tokens. Mixing them with the all-one vector using a token-constant support weight preserves that mean. Multiplication by $a_{i,k}$ yields the component mean identity. The mixture with one remains in $[1-\gamma, 1+\gamma]$, whose lower bound is positive, proving sign preservation. The deviation bound follows from

$$|b_{i,k,t} - a_{i,k}| = |a_{i,k}||w_{i,k,t} - 1| \le \gamma |a_{i,k}|.$$

These statements are algebraic, up to implementation tolerance. They precede objective weighting, batch whitening, PPO clipping, and optimizer preconditioning.

Expanded policy loss and gradient geometry. For a fixed sampled minibatch, the empirical component loss in Equation 9 is

$$\mathcal{L}_k(\theta) = -\frac{1}{M_{\mathcal{B}}} \sum_{i,t} m_{i,t} \min\{\rho_{i,t} b_{i,k,t}, \mathrm{clip}(\rho_{i,t}, 1-\delta, 1+\delta) b_{i,k,t}\}\,. \tag{12}$$

It uses the same global-token-mean reduction as Equation 2, but evaluates one sampled batch rather than taking its expectation. Clipping is applied separately to each component, before gradient combination. At $\theta = \theta_{\mathrm{old}}$, where the surrogate is unclipped, collect valid-token score-function rows $\big(\nabla_\theta \log \pi_\theta(y_{i,t} \mid x_i, y_{i,<t})\big)^\top$ in J and the corresponding token coefficients in b_k. With the coefficients detached,

$$g_k = -J^\top b_k / M_{\mathcal{B}}, \qquad H_{kl} = b_k^\top J J^\top b_l / M_{\mathcal{B}}^2. \tag{13}$$

Credit-space inner products omit $JJ^\top$, the model's score-function geometry, and need not give the same MGDA weights.

MGDA's local minimum-norm property. Let d minimize squared norm over the convex hull of the included g_k. For every vertex g_k, the segment $d + \alpha(g_k - d)$ is feasible for $\alpha \in [0, 1]$. Its right derivative at zero must be nonnegative: $2d^\top(g_k - d) \geq 0$. Therefore $g_k^\top d \geq \|d\|^2$. For differentiable $\mathcal{L}_k$ and nonzero d, $\mathcal{L}_k(\theta - \zeta d) = \mathcal{L}_k(\theta) - \zeta g_k^\top d + o(\zeta)$, which establishes local descent for sufficiently small positive ζ. This proof does not cover arbitrary optimizer transformations, population reward improvement, or variants that alter the direction after solving.

B Implementation and Solver Variants

Reference construction and a two-factor example. The low/high signs encode verified factor levels but are not included in the teacher prompt. Every reference uses the same neutral instruction and must fit the teacher's context together with the prompt and target tokens. For correctness and serialization efficiency, a two-factor example crosses correct or incorrect semantic calls with compact or verbose serialization. Its four references are a correct compact call, a correct verbose call, an incorrect compact call, and the same incorrect call in verbose form. The incorrect calls remain schema-valid. The editor holds non-target reward values fixed within each pair and, where length may confound correctness, matches token counts across correctness levels within each length stratum. The correctness contrast averages the paired differences in the compact and verbose strata. This is an illustration with $K = 2$, not a restriction of the general K-component construction in Section 4.1.

A complete panel is accepted only when its editors and verifiers support every claimed crossing. Structurally coupled rewards may preclude this: if length reward is always zero for incorrect answers, an incorrect/high-length-reward cell may not exist. Such panels receive uniform credit. Verification controls the measured reward factors, not every linguistic property of the edited references.

Allocation support. The support weight $s_{i,k}$ reflects panel validity and the configured transport scale, not an estimated probability that the teacher is correct. It is constant across a component's valid tokens, with $s_{i,k} = 0$ for invalid panels. The component-credit budget is the response-wise mean of the credit coefficients, not a bound on the eventual parameter update.

Batch indexing and token masks. The expectation in Equation 2 is over rollout batches formed by sampling prompts from $\mathcal{D}$ and G responses per prompt from $\pi_{\theta_{\text{old}}}$. Responses are indexed by i across the batch, but advantage normalization uses only responses to the same prompt x_i. The binary mask $m_{i,t}$ is one for optimizable assistant tokens and zero for padding and tool observations. Tool observations may remain in the model context without receiving policy credit. The valid-token counts are $T_i = \sum_t m_{i,t} > 0$ and $M_{\mathcal{B}} = \sum_i T_i$. The latter normalizes the entire batch inside the expectation, rather than weighting each response equally. Sampled tokens, masks, advantages, and the rollout policy parameters θ_{old} are fixed during actor updates.

One training update.

1. Sample student responses and score each reward component. Compute within-prompt advantages using Equation 3.
2. Validate the complete hint panel and score the same student tokens under each accepted reference with a frozen initial-policy teacher.
3. Compute the factorial contrasts and constrained allocation. Invalid panels receive uniform credit; padding receives zero credit.
4. Backpropagate each clipped component loss on the same global minibatch, retaining its parameter-gradient shards.
5. Form the parameter Gram matrix, solve the simplex problem, and combine the saved gradients using detached coefficients.
6. Apply the configured optimizer and log credit invariants, panel coverage, solver residuals, and gradient geometry.

Inference uses only the student and does not require hints or the teacher.

Distributed gradients and normalization. The FSDP implementation accumulates each objective's gradient shards and reduces their inner products to form H. All components share the global minibatch and the loss normalization. The parameter branch selects component credits directly; it does not use the combined-advantage whitening of the single-loss branch. Any KL or entropy term must be specified separately, including whether it participates in the Gram matrix or is added after gradient combination. Mean-credit conservation precedes these choices, loss clipping, and optimizer preconditioning.

Solving the simplex problem. For two objectives, the raw-gradient solution is

$$\lambda_1^* = \mathrm{clip}_{[0,1]} \frac{H_{22} - H_{12}}{H_{11} + H_{22} - 2H_{12}}, \qquad \lambda_2^* = 1 - \lambda_1^*.$$

A symmetric choice handles coincident gradients. For larger objective sets, the implementation uses a simplex Frank–Wolfe solver; its tolerance and residual are part of the run record. Objectives without eligible samples are excluded. Numerical floors that additionally remove weak gradients change the objective set and must be reported.

Normalized and anchored variants. Unit-gradient MGDA replaces g_k by g_k/s_k, with $s_k = \|g_k\|$ for retained nonzero gradients, solves the corresponding Gram problem, and maps coefficients back to the raw gradients as

$$c_k = \frac{\lambda_k/s_k}{\sum_{j\in\mathcal{A}} \lambda_j/s_j}, \qquad d_{\mathrm{unit}} = \sum_{k\in\mathcal{A}} c_k g_k.$$

The positive scalar normalization preserves the direction but changes its magnitude relative to the unnormalized mapping. It is not the raw-gradient solver. A relative norm floor excludes numerically weak objectives and changes the objective set to which any property applies. The implemented anchored family can retain fixed weights outside a conflict regime, invoke MGDA in conflict, and apply anchor correction or skip rules. Any correction must be audited against *all* objectives; protecting an anchor alone does not imply common descent. These variants are distinct from the raw-gradient rule defined in Section 4.3; their guarantees cannot be interchanged. Finite steps, clipping kinks, approximate solves, stochastic sampling, Adam preconditioning, and weight decay do not inherit a general population-reward improvement guarantee.

C Task and Evaluation Details

Three-domain training configuration. The main table uses separate Math, Code, and IF policies initialized from the same Qwen3 checkpoint within each size block. Each domain uses 2,000 training examples and 32 held-out examples. Training uses a learning rate of 10^{-6}, 80 prompts per batch, eight responses per prompt, and three epochs (75 updates). The training prompt and response limits are 2,048 and 8,192 tokens. Training data, prompt templates, reward code, and decoding configuration must match across methods within a domain. Base uses the common initialization; trained methods use the fixed 75-update endpoint rather than the best checkpoint on the benchmark.

Three-domain evaluation configuration. Math evaluation uses temperature 0.6, top-p 0.95, top-k 20, a 30,000-token response limit, and a 32,768-token context limit. Code and IF use temperature 1.0, top-p 0.95, no top-k truncation, and an 8,192-token response limit, with context limits of 16,384 and 32,768 tokens, respectively. Decoding settings match across Base, GRPO, GDPO, GD^{2}PO, and FACTOR-OPD within each domain. All evaluations disable thinking. Table 2 reports arithmetic means over repeated evaluations; the sample counts below refer to one complete evaluation. Math evaluations have 16 generations per problem, giving 480 responses for each complete 30-problem AIME 2024 and AIME 2025 set. Code evaluations have eight generations for every problem: 1,336 generations for the 167-problem LCB v5 subset and 1,400 for the 175-problem LCB v6 subset. A mean correctness score across these generations estimates per-sample accuracy; it is not best-of-eight success. IF uses one sampled response for each of the 541 IFEval and 300 IFBench prompts, with temperature 1.0 rather than greedy decoding. It uses strict prompt-level verification; loose prompt- and instruction-level rates are not the metrics in Table 2. This differs from the original IFBench study, which reports prompt-level loose accuracy and generally uses greedy decoding (Pyatkin et al., 2025).

Reward definitions in the non-thinking setting. Each training example has correctness and format scores, with scalar reference reward $0.8r_{\text{correct}} + 0.2r_{\text{format}}$. Math correctness uses answer equivalence, and its format check requires a boxed answer. Code correctness requires passing the selected execution tests, while its format check requires a nonempty fenced Python block. IF correctness requires all specified constraints to pass. The fractional constraint-satisfaction ratio is a diagnostic, not a replacement for strict correctness. The IF format score checks that the response does not generate thinking tags; the prompt template already provides an empty thinking prefix.

Verifier compatibility. A valid correctness–format panel requires both component verifiers to evaluate every crossed reference. The factors are answer equivalence and boxed-answer formatting for Math, test correctness and code-fence formatting for Code, and constraint satisfaction and thinking-tag formatting for IF. If changing the format makes correctness unmeasurable, that reference cannot represent a valid crossed cell. Incomplete or unverifiable panels receive uniform credit, as specified in Section 4.1. These validity conditions do not imply full panel coverage in every domain; coverage and fallback rates are not reported here.

Checkpoint selection and comparison scope. The main comparison fixes the trained-policy endpoint at update 75; the tool-calling reference protocol uses update 200 (Appendix E). Matching prompt schedules controls data exposure, not the responses produced by diverging on-policy models. The reported score differences are descriptive comparisons, not statistical-significance or population-level Pareto guarantees.

D Ablation Controls

Seeds and aggregation. Both ablations in Section 5.3 use training seeds 42, 43, and 44 for every configuration. Tables 3 and 4 report the arithmetic mean of the three seed-level domain scores. Each domain score is the unweighted mean of its two benchmark accuracies. The configurations use matched prompt schedules, update budgets, and evaluation settings within each domain. The tables report mean scores, not across-seed uncertainty intervals. The interaction estimates from Table 3 are interpreted descriptively, without a statistical-significance claim.

Matched component losses. Table 7 crosses the two algorithmic stages. All four cells use the same per-component clipped losses, normalization, optimizer, and non-policy regularization, with matched prompt and minibatch schedules. Within each run, component losses share the same sampled global minibatch; responses can differ across on-policy runs. For the correctness–format setting, the fixed-gradient controls use coefficients $(0.8, 0.2)$ on the separately clipped component losses, without combined-advantage whitening.

Table 7: Matched mechanism controls. The rows change token allocation; the columns change parameter-gradient combination.

Token allocation	Fixed combination	Parameter MGDA
Uniform	Parameter-fixed control	Outcome-only MGDA
Factorial transport	Transport + parameter-fixed	FACTOR-OPD

For a pre-specified metric S, the allocation–combination interaction is

$$\begin{aligned}\Delta_{\text{int}} = &[S(\text{transport}, \text{MGDA}) - S(\text{uniform}, \text{MGDA})] \\ &- [S(\text{transport}, \text{fixed}) - S(\text{uniform}, \text{fixed})].\end{aligned}$$

This point estimate measures complementarity in the reported means; it does not establish a statistically reliable interaction.

Reference and alignment controls. The natural-sibling control in Section 5.3 forms high-minus-low marginal contrasts from other responses to the same prompt, without constructing a balanced factorial panel. Marginal contrasts on the same complete, balanced factorial panel would equal the paired estimator and therefore would not provide a distinct control. Match the frozen teacher, neutral instruction, target exclusion, hint-count and hint-length budgets, allocation hyperparameters,

and loss path. Natural and controlled references can differ in content and coverage despite matched budgets, so this comparison tests the reference design as a whole rather than isolating a single confounding factor.

The permutation control reuses the factorial references and teacher scores, then permutes the final token weights over valid assistant positions. Use one permutation per response, shared across all reward components and held fixed within an actor update. This preserves the component means, bounds, distributions, and credit-space Gram matrix, while changing the weights' correspondence to the sampled tokens; it does not preserve the parameter-gradient Gram matrix. The aligned factorial row is shared between the two ablation tables, giving six distinct configurations in total.

E Protocols for Detailed Analysis

Tool-calling comparison. Table 5 reproduces the three backbone blocks and all baseline values from Table 1 of Liu et al. (2026a). These are published results, not local reruns; the source reports five-run means and standard deviations, with Overall displayed separately at three decimal places. The FACTOR-OPD rows report level-wise correctness and length-reward summaries for the complete method, including parameter MGDA. The FACTOR-OPD aggregate means are reconstructed from the rounded component means. The processed API-Bank split contains 399, 67, and 131 examples at Levels 1, 2, and 3, respectively, giving $\text{Correct Acc.} = (399a_1 + 67a_2 + 131a_3)/597$, where a_l is the mean percentage accuracy at level l. Overall adds this reconstructed accuracy divided by 100 to the reported mean length reward. These reconstructions may differ in the final displayed digit from aggregates computed using unrounded run-level scores. For FACTOR-OPD, Correct Acc. standard deviations are propagated from the reported level-wise standard deviations s_l as $s_0 = \sqrt{\sum_{l=1}^{3} w_l^2 s_l^2}$, where $(w_1, w_2, w_3) = (399, 67, 131)/597$. This zero-cross-level-covariance approximation gives 1.97, 1.28, and 2.95 percentage points for the three backbones, respectively. These are assumption-based propagated estimates, not empirical across-run standard deviations. The latter require paired run-level scores to account for cross-level covariances.

The reference training protocol is the source's correctness–length setting: processed RLLA-4K training data with the released prompt-length filter, prompt batches of 512, four rollouts per prompt, mini-batches of 128, a learning rate of 10^{-6}, and prompt/response limits of 2,048/1,024. Training sampling uses temperature 1.0 and top-p 1.0. The comparison endpoint is update 200, as specified in the source's Appendix A.3. Correctness ranges from -3 to 3; length scores use the thinking-block word count with the released 512-word normalization, rounding, and clipping. Format is not a third optimized component. Evaluation uses the released processed API-Bank examples, static tool-name/argument matching, and the official generation and scoring scripts, whose default response and context caps are both 4,096 tokens. The baseline source reports five runs without specifying the repetition unit. We interpret the comparisons descriptively rather than as paired comparisons or statistical-significance tests.

Reward-component evaluation. Table 6 compares Qwen3-4B policies on the domain-specific held-out Math and Code splits with the correctness and format verifiers used for training, defined in Appendix C. It reports the final component scores after full training for GRPO, GDPO, GD^{2}PO, and FACTOR-OPD. Correct and Format are percentage scores for the corresponding verifier outcomes, reported separately from the external benchmark accuracies in Table 2.

F Implementation Checks and Evidence Scope

Numerical checks concern credit invariants, uniform fallback, target exclusion, finite teacher scores, masking, distributed gradients, and simplex feasibility; they do not establish benchmark gains. FACTOR-OPD adds teacher scoring and per-component backward passes during training but uses only the student at inference. Equal-GPU-hour gains are not evaluated.